Operating Systems and Languages Library

MS GW-BASIC INTERPRETER per MS-DOS

Guida Utente

olivetti
PERSONAL
COMPUTER

olivetti

Operations Library

MS-DOS

Operazioni fondamentali

Guida all'uso

olivetti
PERSONAL
COMPUTER

olivetti

PREFAZIONE

Questa pubblicazione si rivolge agli utenti di Personal Computer Olivetti che intendono usare il sistema operativo MS-DOS unicamente per eseguire pacchetti applicativi.

Si presume che l'utente abbia letto quelle parti della "Guida all'Installazione e all'Uso" relative al proprio computer, che trattano di setup, operatività e manutenzione del sistema, interfacce modulo base e layout di tastiera.

PRE-REQUISITI: Guida all'Installazione e all'Uso

RIFERIMENTI: MS-DOS Guida Utente

DISTRIBUZIONE: Generale (G)

PRIMA EDIZIONE: Solo in Lingua Inglese

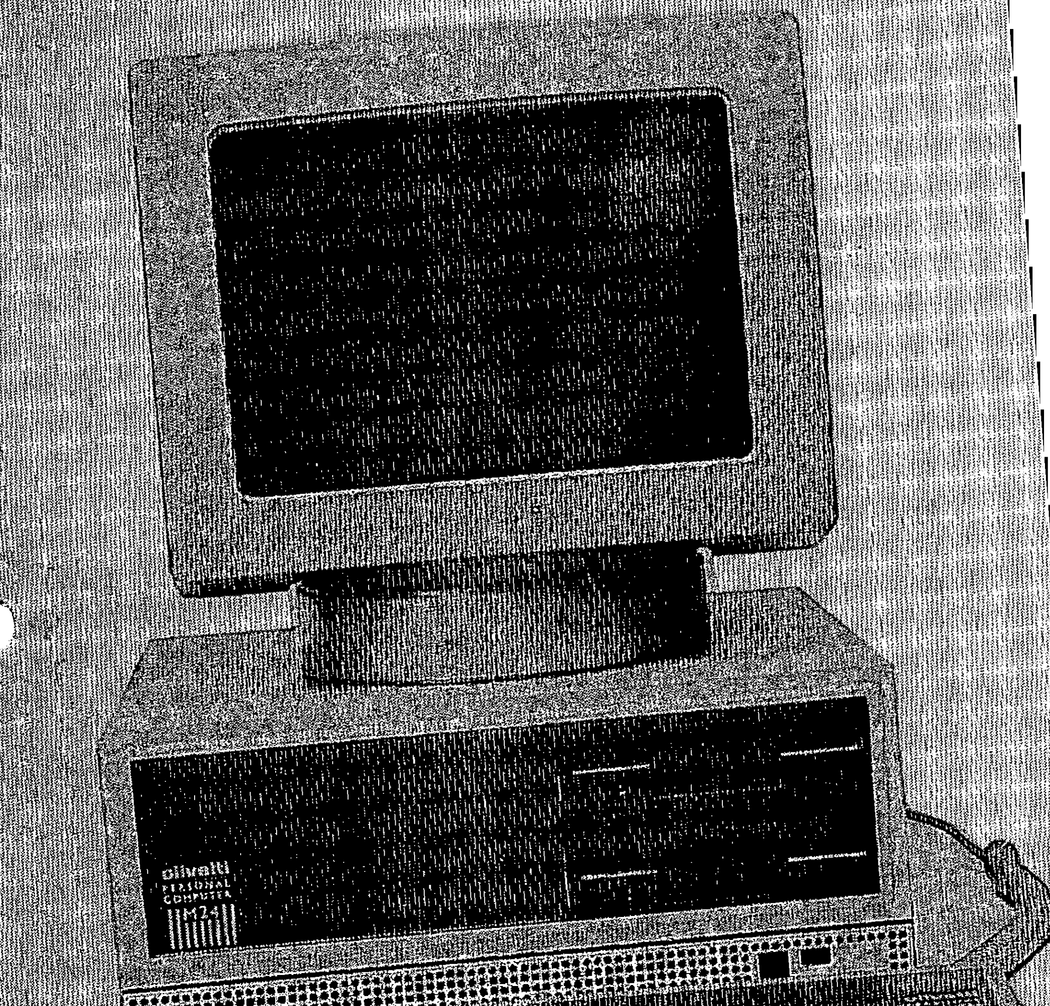
SECONDA EDIZIONE: Settembre 1984

TERZA EDIZIONE: Giugno 1985

QUARTA EDIZIONE: Agosto 1986

PUBBLICAZIONE EMESSA DA:

Ing. C. Olivetti & C., S.p.A.
Direzione Documentazione
77, Via Jervis - 10015 Ivrea (Italy)


olivetti
PERSONAL
COMPUTER
M24

- ASM-86 è un marchio della Digital Research
- CB-86 è un marchio della Digital Research
- CBASIC-86 è un marchio della Digital Research
- CLEO è un marchio della Phone 1 Inc.
- Concurrent CP/M-86 è un marchio della Digital Research
- Concurrent DOS è un marchio della Digital Research
- CP/M-86 è un marchio della Digital Research
- DDT-86 è un marchio della Digital Research
- Dr. Logo è un marchio della Digital Research
- ETHERNET è un marchio della Xerox Corp.
- GSX-86 è un marchio della Digital Research
- GW è un marchio della Microsoft Corp.
- IBM è un marchio registrato della International Business Machines Corp.
- MICROSOFT è un marchio registrato della Microsoft Corp.
- MS è un marchio della Microsoft Corp.
- OMNINET è un marchio della Corvus Systems Inc.
- p-System è un marchio della Softech Microsystem, Inc.
- PC-DOS è un marchio della International Business Machines Corp.
- PEACHPAK è un marchio della Peachtree Software International Ltd.
- Personal Basic è un marchio della Digital Research
- SID-86 è un marchio della Digital Research
- UCSD e UCSD Pascal sono marchi registrati della Regent of the University of California
- UNIX è un marchio registrato della Bell Laboratories
- Z80 è un marchio registrato della Zilog Inc.
- Z8000 è un marchio registrato della Zilog Inc.

INDICE

1. USO DELL'MS-DOS

INDICE

1. USO DELL'MS-DOS

DESCRIZIONE DEL CAPITOLO

Questo capitolo descrive come caricare il sistema operativo MS-DOS. Descrive inoltre i comandi necessari per eseguire le principali operazioni relativamente ai dischetti e all'hard disk.

INDICE

INTRODUZIONE

Per poter usare il sistema è necessario caricare in memoria un sistema operativo.

CHE COSA È UN SISTEMA OPERATIVO

Un sistema operativo è il canale di comunicazione tra voi e il computer. Voi comunicate al sistema operativo le istruzioni che dovranno essere eseguite dal computer, ed esso tradurrà tali istruzioni in una forma comprensibile al computer.

Per comunicare al sistema operativo le istruzioni che il computer dovrà eseguire, introdurrete dei comandi. Ciascun sistema operativo ha un proprio set di comandi.

Questo capitolo è un'introduzione al sistema operativo MS-DOS e descrive alcuni comandi MS-DOS senza però entrare in dettagli. Per maggiori informazioni su formato, regole di sintassi ed opzioni vi rimandiamo al manuale "MS-DOS Guida Utente".

CORREZIONE DEGLI ERRORI DI DIGITAZIONE

È importante ricordare che i comandi devono essere introdotti esattamente nella forma indicata nel testo. Comandi introdotti erroneamente possono provocare la visualizzazione di un messaggio di errore. In questo caso potete tentare di reintrodurre il comando correttamente. Se l'errore persiste, può trattarsi di un problema di dischetti oppure del sistema stesso. Al fine della soluzione di questo problema vi consigliamo di consultare prima il manuale "MS-DOS Guida Utente" quindi, se questo non vi è di aiuto, di contattare il vostro concessionario Olivetti.

PER L'UTENTE DELL'HARD DISK

Il Capitolo 2 descrive come trasferire l'MS-DOS sull'hard disk. Dopo questo trasferimento potrete eseguire molte operazioni direttamente sull'hard disk senza ricorrere ai dischetti. Abbiamo detto molte, ma non tutte; rimangono infatti alcune operazioni che richiedono comunque l'uso dei dischetti, come per esempio:

- Trasferire nuovo software da dischetto su hard disk. (Il nuovo software Olivetti viene distribuito su dischetto.)
- Copiare informazioni esistenti per ragioni di sicurezza, oppure a scopo di distribuzione. (La normale procedura è quella di copiare da hard disk su dischetto.)

A tale scopo vi raccomandiamo di familiarizzare con l'uso dei dischetti.

CARICAMENTO DELL'MS-DOS

L'MS-DOS risiede sul dischetto di sistema, come illustrato nella Figura 1-1.

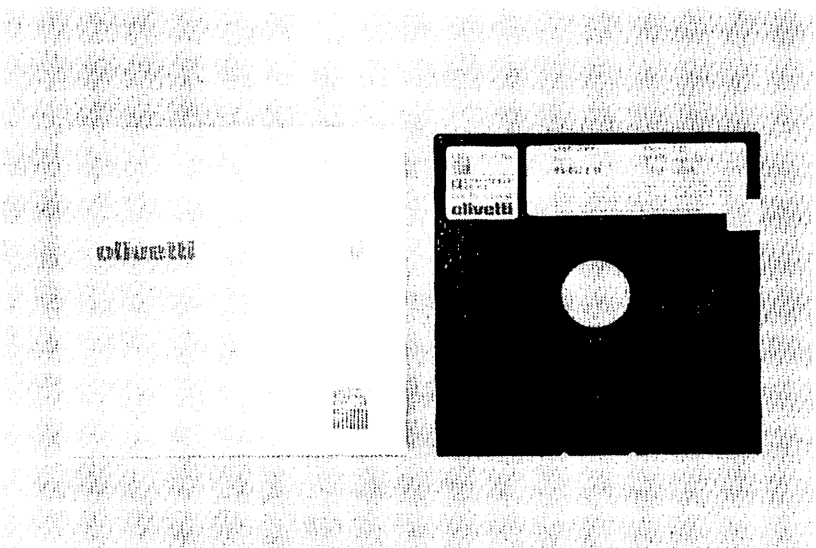


Fig. 1-1 Dischetto di Sistema MS-DOS

Questo dischetto deve già essere protetto da scrittura ed il sistema deve essere acceso, come descritto nel manuale "Guida all'Installazione e all'Uso".

Inserite il dischetto di sistema MS-DOS nel drive e premete un tasto qualsiasi. Appare un messaggio simile al seguente:

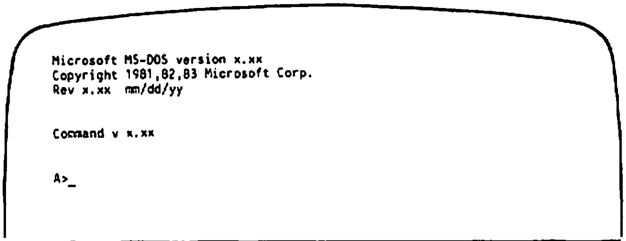


Fig. 1-2 Schermo Video dopo il Caricamento dell'MS-DOS

IL PROMPT DELL'MS-DOS

Il prompt dell'MS-DOS:

A>_

indica che l'MS-DOS è pronto a ricevere i comandi.

DEFINIZIONE DELLA TASTIERA E VERSIONI NAZIONALI

Il sistema che avete ricevuto è configurato per riconoscere la versione di tastiera US ASCII. Se avete una versione diversa e usate il sistema operativo MS-DOS, **DOVETE** eseguire la seguente procedura per indicare al sistema la vostra versione nazionale di tastiera.

1. Riferitevi alla tabella che segue per conoscere il nome del driver corrispondente alla vostra versione nazionale di tastiera.

VERSIONE NAZIONALE	NOME DEL DRIVER	NOME DEL FONT
Danimarca	KEYBDA	NORDIC
Francia	KEYBFR	GRAFTABL
Germania	KEYBGR	GRAFTABL
Grecia	GREEK	—
Italia	KEYBIT	GRAFTABL
Norvegia	KEYBNO	NORDIC
Portogallo	KEYBPO	PORTUGAL
Spagna Internazionale	KEYBSP	GRAFTABL
Spagna	SPAIN1	—
Svezia/Finlandia	KEYBFS	GRAFTABL
Svizzera Francese	KEYBSF	GRAFTABL
Svizzera Tedesca	KEYBSG	GRAFTABL
Gran Bretagna	KEYBUK	GRAFTABL

2. Rimuovete il dischetto di sistema MS-DOS ed inserite il dischetto etichettato **KEYBOARD DRIVERS + UTILITIES** nel drive A.

3. Introducete il nome del driver corrispondente alla vostra tastiera.

4. Quando appare il prompt A, introducete il nome del font corrispondente alla vostra tastiera. Questa procedura è necessaria con MS-DOS release 3.10, mentre con MS-DOS release 2.11 il font si carica automaticamente. (Per i Personal Computer Olivetti M24, M24-3270 ed M24 SP, il caricamento del font non è necessario se non in presenza di tastiera versione Danese, Norvegese o Portoghese).

Avete così definito la versione nazionale della vostra tastiera. Ma, se non caricate il driver (e il font, se necessario) su un file di inizializzazione (AUTOEXEC.BAT), questa procedura deve essere eseguita ogni volta che accendete o inizializzate il sistema. Per creare il suddetto file, seguite le istruzioni che seguono.

CREAZIONE DEL FILE AUTOEXEC.BAT PER IL DRIVER DI TASTIERA E IL FONT

Accertatevi che il vostro dischetto **non** sia protetto da scrittura e, con il prompt

A>_

visualizzato, introducete:

copy con: autoexec.bat

quindi premete **CR** . Se non è richiesto il font, introducete il nome del driver richiesto e premete **CTRL** + **Z** seguito da **CR** .

Se invece è richiesto il font, premete **CR** dopo il nome del driver, quindi introducete il nome del font. Ora premete **CTRL** + **Z** seguito da **CR**

Il file AUTOEXEC.BAT sarà creato e memorizzato sul dischetto. Ogni volta che questo dischetto verrà successivamente usato, il file sarà eseguito, e la vostra versione di tastiera sarà automaticamente riconosciuta dal sistema. Per evitare accidentali sovrascritture sul dischetto, proteggetelo come descritto in precedenza.

NOTA: Le procedure sopra descritte, per la definizione della versione nazionale di tastiera e la creazione del file AUTOEXEC.BAT, richiedono l'uso del dischetto **KEYBOARD DRIVERS + UTILITIES**. Questo dischetto deve essere conservato, poichè è l'unico dischetto contenente un set completo dei driver di tastiera nazionali. Il file AUTOEXEC.BAT deve essere copiato dal dischetto **KEYBOARD DRIVERS + UTILITIES** sull'**MS-DOS SYSTEM DISKETTE** usando il comando COPY descritto in questo capitolo. Dopo aver fatto questo, archiviate il dischetto KEYBOARD DRIVERS + UTILITIES in un posto sicuro e usate l'**MS-DOS SYSTEM DISKETTE** recentemente configurato.

RICHIAMO DEL DRIVER US ASCII

È possibile passare da un driver di tastiera nazionale al driver standard US ASCII premendo **CTRL** + **ALT** + **F1** . Per ritornare al driver della vostra versione nazionale di tastiera, premete **CTRL** + **ALT** + **F2** .

TASTI MULTI-CARATTERE

Su alcune versioni nazionali di tastiera, è presente un certo numero di tasti contrassegnati da più di due simboli.

In rapporto alla posizione del simbolo sul tasto, è possibile una specifica combinazione con altri tasti speciali che devono essere premuti e tenuti abbassati prima di premere il tasto particolare desiderato.

Cinque sono le posizioni possibili su ogni tasto contrassegnato da più di due simboli, anche se non tutte necessariamente riscontrabili sulla totalità di tali tasti.

Posizione 1 - Su un tasto contrassegnato sopra da tre o più simboli, corrisponde al simbolo riportato nell'angolo in basso a sinistra del tasto.

Viene selezionato premendo semplicemente il tasto.

Posizione 2 - Ancora, per lo stesso tipo di tasto, corrisponde al simbolo riportato nell'angolo in alto a sinistra del tasto.

Viene selezionato premendo **SHIFT** + **TASTO** .

Posizione 3 - Corrisponde al simbolo riportato nell'angolo in basso a destra del tasto.

Viene selezionato premendo **ALT** + **TASTO** .

Posizione 4 - Corrisponde al simbolo riportato nell'angolo in alto a destra del tasto.

Viene selezionato premendo **ALT** + **SHIFT** + **TASTO** .

Posizione 5 - Corrisponde al simbolo riportato sulla parte frontale di alcuni tasti.

Viene selezionato premendo **ALT** + **CTRL** + **TASTO** , ad esclusione della tastiera versione Spagna (nome del driver di tastiera: SPAIN1).

In questo caso, il simbolo riportato sulla parte frontale di un tasto viene selezionato premendo **ALT** + **SHIFT** + **TASTO** .

Tasti Multi-carattere

La posizione di questi cinque caratteri sul tasto è illustrata nella figura che segue.

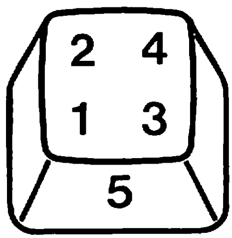


Fig. 1-3 Posizione dei Cinque Caratteri sul Tasto Multi-carattere

TASTIERA SVIZZERA TEDESCA/SVIZZERA FRANCESE

La tastiera versione svizzera supporta le esigenze sia della Svizzera Francese che della Svizzera Tedesca. Tuttavia, per usare la tastiera in un modo o nell'altro deve essere caricato l'apposito driver di tastiera. Vi preghiamo di notare che sebbene alcuni tasti siano contrassegnati da quattro simboli, soltanto un paio saranno attivi in accordo al driver di tastiera che avete caricato.

Così, se avete caricato il driver della tastiera Svizzera Tedesca, saranno selezionati i due simboli presenti sul lato sinistro del tasto seguendo le istruzioni date per le posizioni simbolo 1 e 2.

Se avete caricato il driver della tastiera Svizzera Francese, saranno selezionati i due simboli presenti sul lato destro del tasto, anche qui seguendo le istruzioni date per le posizioni simbolo 1 e 2.

I driver coinvolgono i seguenti tre tasti, come qui illustrato:

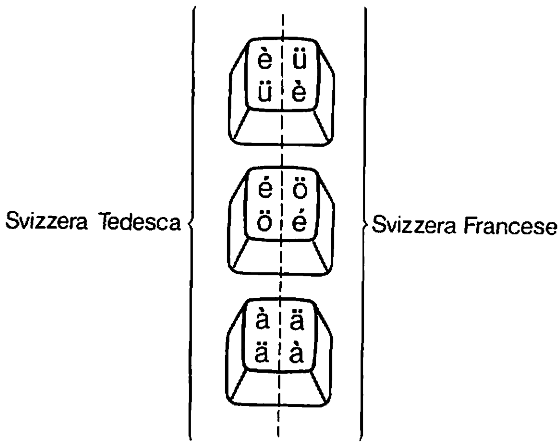


Fig. 1-4

TASTIERA GRECA

La tastiera Greca consente l'introduzione dei caratteri Latini e Greci. All'avviamento la tastiera è settata per introdurre i caratteri Latini. Per passare dall'introduzione dei caratteri Latini a quella dei caratteri Greci, è necessario premere i tasti **ALT** e **CR** . Per ritornare all'introduzione dei caratteri Latini, è necessario premere ancora gli stessi tasti: **ALT** e **CR** .

TASTIERA PORTOGHESE

Sulla tastiera in oggetto i caratteri ü ed Ü vengono generati premendo il tasto ¨ seguito rispettivamente da u ed U.

SIMBOLI MORTI

Alcune versioni nazionali di tastiera supportano determinati simboli utilizzabili soltanto unitamente ad un altro tasto - non possono essere usati da soli. Tali simboli sono definiti **simboli morti**. Essi sono:

´ ` ^ ¨ ~

Questi simboli vengono usati per generare caratteri accentati meno frequentemente usati, non disponibili come caratteri singoli sulla tastiera.

Per ogni versione di tastiera, c'è un set di combinazioni valide di simboli morti. Per generare un carattere accentato, prima deve essere premuto il simbolo morto, quindi l'MS-DOS verifica il successivo carattere introdotto per vedere se è stata introdotta una combinazione valida. Se la combinazione non è valida, il simbolo e il carattere appaiono sullo schermo e il sistema genera un "beep". (Nota: in questo caso il simbolo ¨ appare sullo schermo come ■.)

La tabella che segue riporta le possibili combinazioni simbolo morto/carattere per ciascuna versione nazionale di tastiera.

TASTIERA	SIMBOLI MORTI SUPPORTATI	CARATTERI VALIDI
FRANCIA	¨ ^	ä Ä ë ï ö Ö ü Ü ÿ â ê î ô û
GERMANIA	´ `	á é É í ó ú à è ì ò ù
GRECIA	´ ¨	ά έ ή ί ό ύ ώ ϊ ϋ Ά Έ Ή Ί Ό Ύ Ώ
PORTOGALLO	´ ` ^ ~ ¨	á é í ó ú Á É Í Ó Ú à è ì ò ù À È Ì Ò Ù â ê ô Â Ê Ô ã õ ñ Ã Õ Ñ ü Ü
SPAGNA INTERNAZIONALE, DANIMARCA, SVEZIA/FINLANDIA NORVEGIA	´ ` ^ ¨	á é É í ó ú à è ì ò ù â ê î ô û ä ë ï ö ü ÿ Ä Ö Ü
SPAGNA	´ ` ¨	á é í ó ú Á É Í Ó Ú à è ì ò ù À È Ò ä ë ï ö ü Ä Ï Ö Ü
SVIZZERA FRANCESE SVIZZERA TEDESCA	´ ` ^ ¨ ~	á é í ó ú É à è ì ò ù â ê î ô û ä ë ï ö ü ÿ Ä Ö Ü

TASTI CON CARATTERI INVISIBILI

Le versioni M24 tastiera 2 per Francia, Germania, Italia, Spagna Internazionale e Svizzera supportano caratteri che non appaiono sulla tastiera, ma che possono comunque essere generati. Questi caratteri invisibili e le combinazioni di tasti richieste per generarli sono elencati nella tabella che segue.

NAZIONE	TASTO INVISIBILE	COMBINAZIONE DI TASTI
FRANCIA	|	ALT SHIFT [> < \]
	{	ALT SHIFT [* $ []
	}	ALT SHIFT [& `]]
	~	ALT SHIFT [+ =]
GERMANIA	|	ALT SHIFT [> < \]
	{	ALT SHIFT [* · + []]
	}	ALT SHIFT [^ #]]
	~	ALT SHIFT [` ´]
	'	ALT SHIFT [Ü @]
ITALIA	|	ALT SHIFT [> < \]
	{	ALT SHIFT [& $ []
	}	ALT SHIFT [§ *]]
	~	ALT SHIFT [+ -]
	'	ALT SHIFT [= ì @]
SPAGNA INTERNAZIONALE	|	ALT SHIFT [> < \]
	{	ALT SHIFT [* + []
	}	ALT SHIFT [§ ~]]
	'	ALT SHIFT [P]
SVIZZERA FRANCESE	'	ALT SHIFT [ü è @]
SVIZZERA TEDESCA	'	ALT SHIFT [è ü @]

DEFINIZIONE DELLA DATA E DELL'ORA

Dopo che l'MS-DOS è stato caricato potete, volendo, indicare al sistema la data e l'ora correnti. Esse sono gestite automaticamente dal sistema, anche mentre è spento.

L'MS-DOS usa i valori relativi alla data e all'ora per datare i nuovi file creati quando vengono salvati su disco.

IL COMANDO DATE

Per definire la data introducete:

date

Verrà visualizzata la data, così come è nota al sistema, seguita da una richiesta di introduzione di una nuova data:

Current date is Fri 4-19-85
Enter new date:

Se la data visualizzata è valida, premete semplicemente **CR** .

Per definire una nuova data, introducete la data corrente nella forma:

mese/giorno/anno

Le introduzioni relative a mese, giorno, e anno possono essere separate con barre (come sopra indicato) o con trattini.

Se introducete una data non valida o usate un separatore non valido, viene visualizzato un messaggio di errore insieme con una nuova richiesta di introduzione data.

IL COMANDO TIME

Per definire l'ora introducete:

time

Verrà visualizzata l'ora, così come è nota al sistema, seguita da una richiesta di introduzione di una nuova ora:

Current time is 15:45:00.00
Enter new time:

Se l'ora visualizzata è valida, premete semplicemente **CR** .

Per definire una nuova ora, introducete l'ora corrente nella forma:

ora:minuti:secondi.centesimi di secondo

Se non completate l'introduzione per la nuova ora, i valori rimanenti saranno settati a zero. Per esempio, se introducete **9:15** l'ora sarà settata a **9:15:00.00**

Nota: Se il dischetto di sistema che state usando non contiene il file AUTOEXEC.BAT, i messaggi relativi alla DATA e all'ORA appariranno automaticamente ogni volta che viene caricato l'MS-DOS.

PREPARAZIONE DI UN DISCHETTO

Il dischetto di sistema è già stato formattato ed è quindi pronto per essere usato.

Un dischetto nuovo per essere usato deve essere formattato.

CHE COSA È LA FORMATTAZIONE E PERCHÈ È NECESSARIA

Un dischetto nuovo ha una superficie magnetica (in mylar) sulla quale è possibile definire le posizioni delle tracce e dei settori, per poter memorizzare e reperire informazioni.

Il processo di formattazione definisce le posizioni delle tracce e dei settori sul dischetto e quindi controlla che le informazioni vengano memorizzate e ricercate correttamente. In questo modo qualsiasi difetto nel rivestimento in mylar viene rilevato prima della registrazione delle informazioni sul dischetto stesso.

Se vengono rilevati errori, le aree coinvolte non possono essere usate per la memorizzazione dei dati. Ciò riduce la capacità di memoria del dischetto. Se la capacità massima di memoria presenta delle criticità per quanto concerne le vostre operazioni, in caso di riscontro di aree di memoria difettose rieseguite l'utility format. Se l'errore si ripete, formattate un nuovo dischetto.

Oltre a definire le posizioni delle tracce e dei settori il processo di formattazione dell'MS-DOS crea sul dischetto una directory di sistema.

LA DIRECTORY DI SISTEMA

La directory di sistema è l'indice dei file contenuti in un dischetto. Senza la directory l'MS-DOS non sarebbe in grado di conoscere quali dati sono stati registrati sul dischetto, dove sono stati memorizzati tali dati e quali aree sono ancora libere per la registrazione di nuovi dati. (Nota: Per riciclare vecchi dischetti già formattati per MS-DOS, non è necessario definire nuove tracce. È sufficiente azzerare la directory usando un'opzione nell'ambito del comando FORMAT, come descritto nel manuale "MS-DOS Guida utente".)

COME FORMATTARE CON SISTEMA VERSIONE A DUE DRIVE

Digitate:

format b:

Appare un messaggio simile al seguente:

Insert new diskette for drive B:
and strike any key when ready

Inserite il dischetto che desiderate formattare nel drive superiore e premete **CR** .

Ha così inizio il processo di formattazione durante il quale viene visualizzato un messaggio.

Al termine di questo processo vengono visualizzate alcune informazioni relative allo spazio sul dischetto ed appare il seguente messaggio:

Format another (Y/N)?

Digitate N e riapparirà il prompt dell'MS-DOS.

COME FORMATTARE CON SISTEMA VERSIONE AD UN SOLO DRIVE

Digitate:

format a:

Appare un messaggio simile al seguente:

Insert new diskette for drive A:
and strike any key when ready

Rimuovete il dischetto di sistema e sostituitelo con il dischetto che desiderate formattare. Premete **CR** .

Inizia così il processo di formattazione e viene visualizzato un messaggio. Al termine di questo processo vengono visualizzate alcune informazioni relative allo spazio sul dischetto ed appare il seguente messaggio:

Format another (Y/N)?

Rimuovete il dischetto appena formattato e sostituitelo con il dischetto di sistema. Digitate N e riapparirà il prompt dell'MS-DOS.

Per ulteriori informazioni sulle opzioni di formattazione, vi rimandiamo al manuale "MS-DOS Guida utente".

Nota: Per formattare un dischetto da usare come dischetto di sistema, dovete usare il seguente formato:

format n:/s

dove n è l'appropriato identificatore di drive.

VOLUMI

Dopo la creazione della directory di sistema, il dischetto viene comunemente chiamato volume. L'MS-DOS consente di assegnare ai volumi un nome. Per esempio, se desiderate tenere tutte le informazioni relative alle paghe su un dischetto, e tutte le informazioni relative alla gestione di STOCK di magazzino su un altro dischetto, potete assegnare i nomi PAGHE e STOCK ai rispettivi dischetti. L'assegnazione del nome ad un volume può essere effettuata nel momento della formattazione del dischetto. Da quel momento, quel nome apparirà ogni volta che richiederete il listing della directory di sistema del volume in oggetto.

FILE

Le informazioni vengono memorizzate su un volume in unità dette file. Schematizzando, i file possono essere di due tipi: file programma e file dati.

I file programma (comunemente chiamati programmi) possono essere usati per eseguire una elaborazione specifica, quale ad esempio la stampa di una serie di buste paga.

Un file dati è costituito da un insieme di voci correlate tra loro, come ad esempio un elenco di nomi e relativi indirizzi.

I programmi sono creati usando i diversi linguaggi di programmazione che il sistema supporta, e i file dati possono essere creati eseguendo un programma. Potete inoltre creare file dati utilizzando l'editor di linea dell'MS-DOS **edlin** o l'editor di video EDIT. Se siete programmatori probabilmente necessitate di informazioni dettagliate sui linguaggi di programmazione e sugli editor. Vi rimandiamo a tale proposito ai manuali relativi ai vari linguaggi e al manuale "MS-DOS Guida Utente".

NOMI DEI FILE

Ogni file è identificato da un nome. Un nome di file può essere composto:

- da uno a otto caratteri (per caratteri ammessi vedere di seguito)

- da uno a otto caratteri, seguiti da un punto (.) e dall'estensione del nome che può essere composta da uno a tre caratteri.

I caratteri alfabetici per quanto concerne i nomi dei file possono essere introdotti sia in maiuscolo sia in minuscolo, ma l'MS-DOS tradurrà in maiuscolo quelli in minuscolo.

I caratteri ammessi per creare un nome di file sono i seguenti:

A-Z	0-9	-	!
.	'	(	)
&	,	_	%

LE DIRECTORY

L'MS-DOS consente di raggruppare i file in directory. Un volume può contenere diverse directory. La directory primaria di un volume è chiamata "root". La directory root è la directory di sistema che viene creata con il comando FORMAT, come descritto in precedenza in questo capitolo. La figura che segue vi sarà di aiuto nel definire l'uso delle directory.

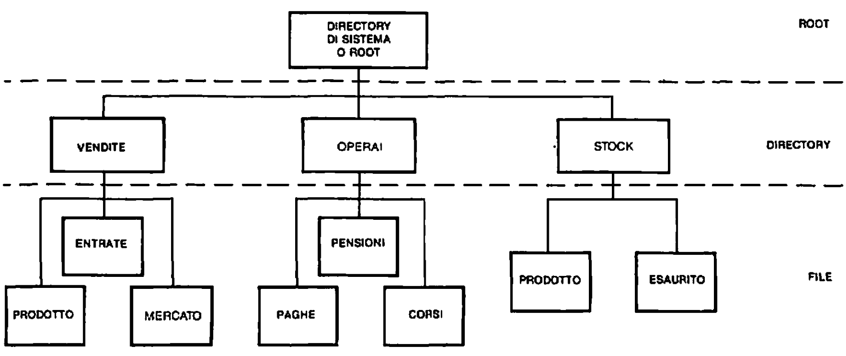


Fig. 1-5 Directory e File

Per accedere a un file in questo volume è necessario definire il "path" (percorso) dalla directory root al particolare file, per esempio:

command \OPERAI\PENSIONI

invierà un comando MS-DOS al file "PENSIONI" memorizzato nella directory "OPERAI". Il primo simbolo \ dice all'MS-DOS che la ricerca deve partire dalla directory root, o directory di sistema. Il secondo simbolo \ è usato per separare il nome della directory dal nome del file.

Con l'uso delle directory è possibile avere file con nomi identici sullo stesso volume. La Figura riporta due file chiamati "PRODOTTO", uno memorizzato nella directory "VENDITE", l'altro nella directory "STOCK". Ciascuno dei due file "PRODOTTO" è accessibile da path diversi. Specificando un unico path dite all'MS-DOS a quale file "PRODOTTO" deve accedere.

Per maggiori informazioni sulle directory e loro uso, vi rimandiamo al manuale "MS-DOS Guida utente".

LISTING DEI VOLUMI

Potete visualizzare i nomi delle directory e dei file memorizzati nella directory di sistema usando il comando DIR. Questo comando vi permette inoltre di avere informazioni circa lo spazio occupato da ciascun file e lo spazio ancora disponibile sul volume.

Per vedere ciò che è memorizzato sul dischetto di sistema digitate dunque:

dir a:

Lo schermo appare simile a quello illustrato nella Figura 1-6.

```
HEXDUMP   COM     567   4-27-85   9.00a
KEYBFR    COM    6916   4-27-85   9.00a
KEYBGR    COM    6916   4-27-85   9.00a
KEYBIT    COM    6916   4-27-85   9.00a
KEYBSF    COM    6916   4-27-85   9.00a
KEYBSG    COM    6916   4-27-85   9.00a
KEYBSP    COM    6916   4-27-85   9.00a
KEYBFS    COM    6916   4-27-85   9.00a
KEYBNO    COM    6916   4-27-85   9.00a
KEYBUK    COM    6916   4-27-85   9.00a
LINK      EXE   42330   4-27-85   9.00a
MODE      COM    2382   4-27-85   9.00a
MORE      COM    4380   4-27-85   9.00a
PRINT     COM    4588   4-27-85   9.00a
RECOVER   COM    2380   4-27-85   9.00a
RESTORE   COM    4171   4-27-85   9.00a
SORT      EXE    1632   4-27-85   9.00a
SYS       COM    2832   4-27-85   9.00a
TREE      COM    1336   4-27-85   9.00a
KEYBDA    COM    6916   4-27-85   9.00a
NORDIC -  'COM   3287   4-27-85   9.00a
           File(s)      2690 bytes free

A>_
```

Fig. 1-6 Directory di Sistema del Dischetto di Sistema

È facile distinguere tra file e directory in quanto la directory, se presente, ha dopo il nome la definizione **DIR** .

OPERAZIONI SUI FILE

Le varie operazioni sui file riguardano la copia, la cancellazione, l'assegnazione di nuovi nomi, etc. Molte sono le ragioni per le quali si vuole intervenire su un file. Per esempio, per farne una copia di backup, per farne una copia a scopo di distribuzione, oppure per cancellarlo perchè non serve più e quindi recuperarne lo spazio occupato.

Proviamo ad operare sui file facendo quanto segue.

COPIA DI FILE CON SISTEMA VERSIONE A DUE DRIVE

Osservate la directory del dischetto di sistema. Come potete vedere, l'MS-DOS è costituito da numerosi file. Copieremo il file SORT.EXE. Noterete che nella directory il nome compare composto da due parole: SORT ed EXE, separate da uno spazio. EXE è l'estensione del nome del file. Nel listing della directory di sistema, le estensioni dei nomi compaiono allineate verticalmente per facilità di lettura. Nei comandi MS-DOS i nomi dei file sono separati dall'eventuale estensione da un punto.

Per fare la copia del file SORT.EXE usate il comando COPY. Digitate:

copy a:SORT.EXE b:temp

Il file viene in tal modo copiato sul dischetto presente nel drive superiore. Viene visualizzato il seguente messaggio:

1 File(s) copied

e compare il prompt dell'MS-DOS. Come avrete notato per la copia è stato specificato un nuovo nome di file: **temp**

Osserviamo ora la directory del nuovo dischetto.

Digitate:

dir b:

Appare un messaggio simile al seguente:

Volume in drive B: has no label
Directory of B:
TEMP 1632 4-27-85 9:00a
1 File(s) 173056 bytes free

Come potete vedere il dischetto contiene il file chiamato TEMP. È la copia esatta del file SORT.EXE presente sul dischetto di sistema. Il file occupa 1632 byte di memoria.

CANCELLAZIONE DI FILE CON SISTEMA VERSIONE A DUE DRIVE

Per cancellare il file temp dal dischetto usate il comando DEL. Digitate:

del b:temp

Se ora osservate nuovamente la directory del dischetto introducendo:

dir b:

appare un messaggio simile al seguente:

File not Found

Ciò significa che il dischetto è vuoto e che il file **temp** è stato cancellato. Lo spazio occupato dai 1632 caratteri di questo file è così nuovamente disponibile per la memorizzazione di altre informazioni.

COPIA DI FILE CON SISTEMA VERSIONE AD UN SOLO DRIVE

L'esempio che segue descrive come copiare il file SORT.EXE dal dischetto di sistema sul dischetto di lavoro precedentemente formattato. Se digitate:

copy a:sort.exe b:

l'MS-DOS trasferirà in memoria il file SORT.EXE. Una volta che il file è stato trasferito in memoria, l'MS-DOS visualizza il seguente messaggio:

Insert diskette for drive B:
and strike any key when ready

Nota: Anche se il messaggio fa riferimento al drive **b:** intendete che il riferimento del messaggio sia fatto a un dischetto e non a un drive.

A questo punto, rimuovete il dischetto di sistema e sostituitelo con il dischetto formattato. Premete un tasto qualsiasi, quasi immediatamente verrà visualizzato il seguente messaggio:

1 File(s) copied

Il dischetto ora contiene una copia di SORT.EXE come potete sincerarvene introducendo il comando DIR. L'MS-DOS visualizzerà un messaggio simile al seguente:

```
Volume in drive A: has no label
Directory of A:
SORT.EXE  1632        4-27-85       9:00a
1 File(s)       360448 bytes free
```

Come potete vedere il file copiato ha le stesse caratteristiche del file originale: lo stesso nome, la stessa dimensione. Se desiderate che il file copiato abbia un nome diverso dall'originale, specificate semplicemente quel nome dopo l'identificatore **b:** nel comando COPY.

CANCELLAZIONE DI FILE CON SISTEMA VERSIONE AD UN SOLO DRIVE

Per cancellare il file SORT.EXE dal dischetto di lavoro, usate il comando DEL. Con il dischetto di lavoro ancora nel drive introducete:

del a:sort.exe

Se ora osservate nuovamente la directory del dischetto, introducendo:

dir a:

verrà visualizzato un messaggio simile al seguente:

File not found

Ciò significa che il dischetto è vuoto e che la copia di SORT.EXE è stata rimossa. Lo spazio occupato dai 1632 byte è nuovamente disponibile per la memorizzazione di altre informazioni.

FILE PARTICOLARI

Prima di chiudere l'argomento sui file, parleremo di alcuni tra i più importanti fra essi.

Il File di Inizializzazione

È possibile creare un particolare tipo di file detto file di inizializzazione. Se viene creato un file di inizializzazione contenente i comandi dell'MS-DOS, esso viene eseguito automaticamente ogni qualvolta viene caricato l'MS-DOS.

Il file di inizializzazione può essere utilizzato per evitare l'esecuzione di operazioni ripetitive all'inizio di ogni lavoro. Per esempio, se all'accensione del sistema volete che venga sempre eseguito un particolare programma, potete ricorrere al file di inizializzazione.

Se decidete di creare un file di inizializzazione, dovete chiamarlo AUTOEXEC.BAT. Ogni volta che caricate l'MS-DOS, esso cerca un file con questo nome.

Un esempio di creazione di un file AUTOEXEC.BAT è stato dato all'inizio di questo capitolo in merito al caricamento automatico del driver di tastiera.

Nell'ambito del file di inizializzazione possono essere usati speciali comandi MS-DOS, come per esempio:

- il comando REM, che vi consente di scrivere commenti
- il comando PAUSE, che arresta l'esecuzione del file AUTOEXEC.BAT e attende il vostro consenso per continuare, oppure l'indicazione a terminare il programma.

I comandi disponibili per AUTOEXEC.BAT possono anche essere utilizzati per i file di tipo batch dell'MS-DOS.

File di Tipo Batch

Questi file possono essere usati per elaborare processi ripetitivi. Essi non vengono eseguiti automaticamente, ma solamente quando richiesti tramite l'MS-DOS. Per esempio, se sovente avete la necessità di ordinare un file clienti per nome e poi eseguire un programma per la stampa del file, potete creare un file di tipo batch. Esso vi eviterà di introdurre tutti i comandi ogni volta che volete eseguire questo tipo di lavoro.

Per informazioni più dettagliate relative ai file di tipo batch e al file AUTOEXEC.BAT vi rimandiamo al manuale "MS-DOS Guida utente".

COPIA DI UN DISCHETTO

Un'operazione che molto spesso avrete necessità di eseguire è la copia di un dischetto. Questa necessità può essere determinata da varie ragioni; una delle più ovvie è quella di creare una copia di backup. Tratteremo questo argomento più avanti. Vogliamo qui invece richiamare la vostra attenzione su un problema che può sorgere nel duplicare un dischetto contenente software applicativo.

DISCHETTI CONTENENTI SOFTWARE APPLICATIVO

Il software applicativo viene distribuito su dischetto. I dischetti che contengono software applicativo riguardano una vasta gamma di pacchetti: gestione dati, tabelle per il calcolo e l'analisi interattiva dei dati, etc. I dischetti contenenti pacchetti applicativi sono a volte protetti da copia.

PROTEZIONE DA COPIA

La protezione da copia varia da dischetto a dischetto. Per esempio, vi può essere consentito di fare un numero limitato di copie, oppure di copiare da un dischetto soltanto un certo numero di file.

Come regola generale, prima di copiare dischetti contenenti software applicativo vi suggeriamo di consultare la relativa documentazione oppure il contratto di licenza che ne regola l'utilizzo.

CREAZIONE DI COPIE DI BACKUP

Non si ripeterà mai abbastanza quanto importante sia fare copia di backup di dischetti contenenti informazioni vitali. Molti utenti di computer fanno copie di backup dei file che hanno modificato alla fine di ciascuna giornata di lavoro.

A questo punto anche voi siete già in possesso di un dischetto contenente informazioni importanti: il dischetto di sistema MS-DOS. Vi illustriamo qui di seguito la procedura da seguire per fare una copia di backup di questo dischetto.

Nota: L'MS-DOS formatterà automaticamente il dischetto su cui copiare, se non ancora formattato.

COPIA DI BACKUP CON SISTEMA VERSIONE A DUE DRIVE

Per copiare dal dischetto di sistema MS-DOS sul dischetto che avete utilizzato in precedenza per operare sui file, usate il comando DISKCOPY. Digitate:

diskcopy a: b:

Compare un messaggio simile al seguente:

Insert source diskette into drive A:
Insert destination diskette into drive B:
Strike any key when ready

Il dischetto sorgente è il dischetto di sistema MS-DOS, già presente nel drive A. Il dischetto destinatario è quello che avete utilizzato per operare sui file e che è già presente nel drive B. Ricorderete che l'operazione di copia distrugge tutte le informazioni presenti sul dischetto destinatario. Ma poichè il dischetto in questione nel nostro caso non contiene informazioni, potete tranquillamente procedere premendo **CR** .

Al termine dell'operazione di copia viene visualizzato il seguente messaggio:

Copy complete
Copy another disk (Y/N) ?

Digitate **N** ed apparirà il prompt dell'MS-DOS.

Avete così creato una copia di backup del dischetto di sistema. Vi consigliamo ora di mettere il dischetto di sistema originale in un posto sicuro e di continuare a lavorare con la copia. Prima di usare la copia di lavoro vi suggeriamo di proteggerla ed etichettarla. Se la copia di lavoro dovesse diventare inutilizzabile, potrete ricavare un'altra copia dall'originale.

COPIA DI BACKUP CON SISTEMA VERSIONE AD UN SOLO DRIVE

Per copiare il dischetto di sistema sul dischetto che avete in precedenza formattato, digitate:

diskcopy

Compare un messaggio simile al seguente:

Insert source diskette into drive A:
Strike any key when ready

Premete **CR** . Il sistema operativo MS-DOS copia nella memoria del sistema quanto più possibile del dischetto. Al termine di questa operazione compare il seguente messaggio:

Insert destination diskette into drive A:
Strike any key when ready

Rimuovete il dischetto di sistema e sostituitelo con il dischetto formattato. Premete **CR** . Quella parte del dischetto di sistema che è stata registrata nella memoria del sistema viene ora registrata sul dischetto che avete appena caricato. Al termine di questa operazione, può comparire il seguente messaggio:

Insert source diskette in drive A:
Strike any key when ready

Ciò significa che il vostro sistema non ha memoria disponibile sufficiente per contenere in una sola volta l'intero contenuto del dischetto di sistema. Pertanto scambiate nuovamente i dischetti e premete **CR** . In questo modo verrà tasferito in memoria, come prima, il seguito del dischetto di sistema.

Questa operazione di scambio dei dischetti deve continuare fino a quando non è stato copiato l'intero contenuto del dischetto di sistema. Ricorderete che il dischetto sorgente è quello dal quale state copiando, mentre il dischetto destinatario è quello sul quale state copiando.

Alla fine viene visualizzato il seguente messaggio:

Copy complete
Copy another disk (Y/N) ?

Digitate **N** ed appare il prompt dell'MS-DOS.

Avete così creato una copia di backup del dischetto di sistema. Vi consigliamo ora di mettere il dischetto di sistema originale in un posto sicuro e di continuare a lavorare con la copia. Prima di usare la copia di lavoro vi suggeriamo di proteggerla ed etichettarla. Se la copia di lavoro dovesse diventare inutilizzabile, potrete ricavare un'altra copia dall'originale.

ALTRE PRESTAZIONI DELL'MS-DOS

Le prestazioni dell'MS-DOS viste finora non sono che una piccola parte di quelle disponibili. Per una visione globale vi rimandiamo al manuale "MS-DOS Guida utente". Altre prestazioni tra le più comunemente usate sono brevemente riassunte nella Tabella 1-7.

COMANDO MS-DOS	DESCRIZIONE
chkdsk	Analizza un dischetto (oppure l'hard disk) controllando la coerenza dei dati e l'eventuale presenza di tracce difettose
sort	Ordina un determinato file in ordine alfabetico, ascendente o discendente
type	Visualizza il contenuto di un determinato file

Tab. 1-7 Altri Comandi MS-DOS

FINE DELLA SESSIONE

Abbiamo terminato le note introduttive all'MS-DOS. Avete ora sufficienti informazioni per gestire efficacemente i vostri file e volumi e siete in grado di iniziare ad eseguire il vostro software applicativo sotto MS-DOS.

2. L'MS-DOS E L'HARD DISK

DESCRIZIONE DEL CAPITOLO

Questo capitolo descrive come preparare l'hard disk per l'uso unicamente da parte dell'MS-DOS oppure congiuntamente ad altri sistemi operativi.

INDICE

INTRODUZIONE

Se il vostro sistema è dotato di hard disk prima di poterlo usare con il sistema operativo MS-DOS dovete seguire una delle procedure sotto indicate:

1. Se l'MS-DOS sarà l'unico sistema operativo che userete, dovete dedicare l'intero hard disk all'MS-DOS, come descritto nella sezione "Come dedicare l'hard disk unicamente all'MS-DOS".

2. Se intendete usare altri sistemi operativi oltre all'MS-DOS, allora dovete ripartire l'hard disk in aree indipendenti dette partizioni. Dovrete avere una partizione per ciascun sistema operativo. Ciascuna partizione dovrà essere definita usando il sistema operativo che utilizzerà quella partizione. Le operazioni da eseguire relativamente all'MS-DOS sono descritte nella sezione "Come ripartire l'hard disk per più di un sistema operativo".

Per informazioni su come predisporre le partizioni non MS-DOS, vi rimandiamo al manuale relativo allo specifico sistema operativo.

NOTA: In questo capitolo parleremo di sistema "bootable". Il sistema bootable è quello che viene caricato automaticamente all'accensione del sistema o dopo l'esecuzione di un reset.

Il comando MS-DOS da usare per predisporre l'hard disk è il comando FDISK. Esso produce una serie di quadri video e di prompt che vi guidano passo a passo nell'esecuzione delle varie operazioni.

COME DEDICARE L'HARD DISK UNICAMENTE ALL'MS-DOS

1. Caricate l'MS-DOS dal dischetto presente nel drive A e digitate:

   fdisk

   Sullo schermo appare il menu delle Opzioni FDISK, simile al seguente:

```
Fixed Disk Setup Program

FDISK Options

Choose one of the following:

1. Create MS-DOS Partition
2. Change Active Partition
3. Delete MS-DOS Partition
4. Display Partition Information

Enter choice .....................: :[ 1 ]

Press Esc to return to DOS ...............[ ]
```

Fig. 2-1

2. Digitate:

   1

   Apparirà un messaggio simile al seguente:

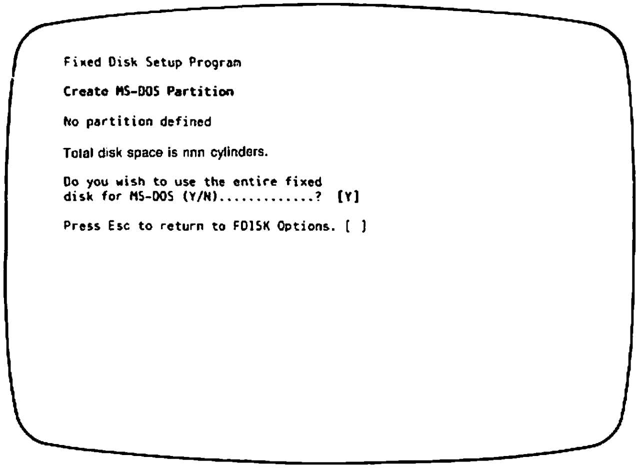


Fig. 2-2

3. Premete:

   CR

   e l'intero hard disk sarà dedicato all'MS-DOS, come indica il messaggio che viene visualizzato sullo schermo.

4. Premete i tasti **CTRL** , **ALT** e, con questi abbassati, premete il tasto **DEL** : l'MS-DOS verrà ricaricato automaticamente. L'MS-DOS riconoscerà l'hard disk come drive valido contenente la partizione MS-DOS. Da questo punto in poi l'hard disk è indirizzabile usando l'identificatore di drive C.

5. A questo punto dovete formattare l'hard disk.

   Se volete che l'MS-DOS venga caricato automaticamente dall'hard disk, digitate:

   format c:/s

Se, viceversa, non volete che l'MS-DOS venga caricato automaticamente dall'hard disk, digitate semplicemente:

format c:

In entrambi i casi appare il seguente messaggio:

Press any key to begin formatting C:

6. Premete un tasto qualsiasi ed appare il seguente messaggio:

   Formatting...

   Dopo alcuni minuti apparirà quest'altro messaggio:

   Format complete

   e se digitate **/s** nella riga comandi apparirà anche il seguente messaggio:

   System transferred

   Ciò significa che l'MS-DOS è stato copiato sull'hard disk. Vi sarà ora possibile effettuare il caricamento del sistema operativo MS-DOS dall'hard disk.

7. Se avete copiato l'MS-DOS sull'hard disk necessiterete di copiare sull'hard disk anche i comandi esterni dell'MS-DOS. Con il dischetto di sistema MS-DOS ancora nel drive A digitate:

   copy *.*,c:

Per verificare se l'MS-DOS è stato trasferito con esito positivo, rimuovete il dischetto MS-DOS dal drive A e premete i tasti **CTRL** , **ALT** e, con questi abbassati, premete il tasto **DEL** per effettuare un reset di sistema. L'MS-DOS dovrà essere caricato dall'hard disk.

COME RIPARTIRE L'HARD DISK PER PIÙ DI UN SISTEMA OPERATIVO

Ogni sistema operativo che supporta l'hard disk ha un comando che consente di definire la partizione che il particolare sistema operativo userà.

Il comando FDISK vi consente di:

- definire la partizione MS-DOS di qualsiasi dimensione (limitatamente alla capacità dell'hard disk) ed in qualsiasi locazione nell'ambito dell'hard disk
- cambiare la partizione che sarà attiva quando il sistema viene ricaricato
- cancellare la partizione MS-DOS
- visualizzare la mappa delle partizioni dell'hard disk.

Per richiamare il programma FDISK digitate:

fdisk

Verrà visualizzato il menu delle Opzioni FDISK come segue:

```
Fixed Disk Setup Program

FDISK Options

Choose one of the following:

1.  Create MS-DOS Partition
2.  Change Active Partition
3.  Delete MS-DOS Partition
4.  Display Partition Information

Enter choice.....................: : [ 1 ]

Press Esc to return to DOS ...............[ ]
```

Fig. 2-3

Procedete introducendo il numero corrispondente alla scelta che intendete fare.

Come Creare la Partizione MS-DOS

1. Fate la scelta 1 nell'ambito del menu delle Opzioni FDISK digitando:

   1

2. Appare il seguente messaggio:

```
Fixed Disk Setup Program

Create MS-DOS Partition

No partition defined

Total disk space is nnn cylinders.

Do you wish to use the entire fixed
disk for MS-DOS (Y/N)............?  [Y]

Press Esc to return to FDISK Options. [ ]
```

Fig. 2-4

3. Digitate:

   n

   Viene visualizzato un messaggio simile al seguente:

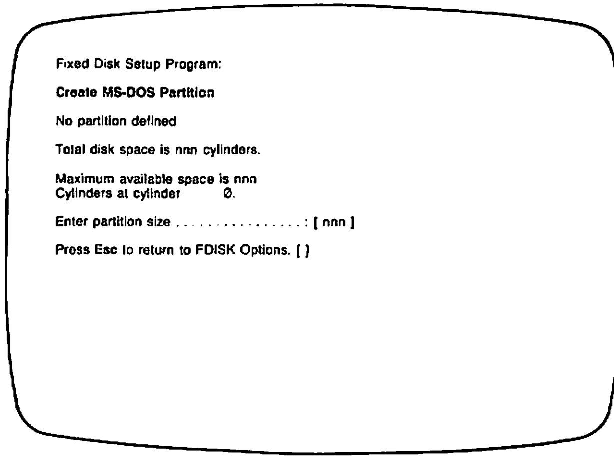


Fig. 2-5

   Da questo messaggio potete rilevare il numero totale dei cilindri, la dimensione e la locazione del massimo numero di cilindri contigui disponibili sull'hard disk. Dal momento che l'hard disk non contiene alcunchè, il numero di cilindri contigui disponibili corrisponde allo spazio totale dell'hard disk.

4. L'FDISK vi propone quanto segue:

   Enter partition size.............:

   Il valore di default è il massimo spazio disponibile. Premete unicamente **CR** se volete accettare questo valore, oppure digitate il numero di cilindri desiderato seguito da **CR** .

   Verrà quindi visualizzato il seguente messaggio:

   Enter starting cylinder number..: [Ø]

5. Il valore di default per il numero del cilindro d'inizio dipende dalla dimensione che avete assegnato alla partizione. È il primo cilindro che occupa minor spazio sull'hard disk in grado di contenere la partizione. Se corrisponde a quello in cui volete la partizione MS-DOS, premete **CR**. Diversamente, digitate il numero del cilindro che volete come primo.

6. Se volete continuare a lavorare con il comando FDISK premete una volta **Esc**. Diversamente, premete i tasti **CTRL**, **ALT** e, con questi abbassati, premete il tasto **DEL** per ricaricare automaticamente il sistema.

7. Dovete ora formattare la partizione MS-DOS in modo che l'MS-DOS possa usarla.

   Se volete che l'MS-DOS venga caricato automaticamente dall' hard disk digitate:

   format c:/s

   Se, viceversa, non volete che l'MS-DOS venga caricato automaticamente dall'hard disk è sufficiente che digitiate:

   format c:

   In entrambi i casi appare il seguente messaggio:

   Press any key to begin formatting C:

8. Premete un tasto qualsiasi ed appare il seguente messaggio:

   Formatting...

   Dopo alcuni minuti apparirà quest'altro messaggio:

   Format complete

   e se digitate **/S** nella riga comandi apparirà anche il seguente messaggio:

   System transferred

   Questo messaggio sta ad indicare che l'MS-DOS è stato copiato sull'hard disk.

9. Se avete copiato l'MS-DOS sull'hard disk necessiterete di copiare sull'hard disk anche la libreria dei comandi esterni MS-DOS. Con il dischetto di sistema MS-DOS ancora nel drive A, digitate:

   copy *.*,c:

   Per rendere caricabile l'MS-DOS dall'hard disk dovete rendere attiva la partizione MS-DOS, come di seguito descritto. Se, invece, non volete che l'MS-DOS venga caricato automaticamente dall'hard disk non avete questa necessità.

DEFINIZIONE DI SISTEMA BOOTABLE

Il sistema bootable è quello che occupa la partizione attiva ed è quello che verrà automaticamente caricato all'accensione del sistema o dopo l'esecuzione di un reset.

L'esempio usato in questa sezione illustra un hard disk con quattro partizioni, ciascuna contenente un diverso sistema operativo. La terza partizione contiene l'MS-DOS che sarà definito come il sistema bootable.

1. Per definire il sistema bootable, dal menu delle Opzioni FDISK, digitate:

   2

   Apparirà un messaggio simile al seguente:

```
Fixed Disk Setup Program

Change Active Partition

Partition   Status     Type        Start   End    Size
    1         N       non DOS       000    044     45
    2         A       non DOS       045    089     45
    3         N         DOS         090    134     45
    4         N       non DOS       135    179     45

Total disk space is xxxx cylinders.
The current active partition is 2

Enter the number of the partition you
want to make active..............:[ ]

Press Esc to return to FDISK Options. [ ]
```

Fig. 2-6

2. Digitate il numero della partizione il cui sistema operativo volete rendere attivo. Per esempio:

   3

   rende attivo il sistema operativo nella partizione 3. Diventa quindi inattivo il sistema operativo nella partizione 2.

3. Per caricare il sistema operativo nella partizione che avete appena resa attiva, rimuovete il dischetto MS-DOS e premete i tasti **CTRL** , **ALT** e, con questi abbassati, premete il tasto **DEL** .

BACKUP DI FILE SU HARD DISK

L'MS-DOS dispone di due comandi specifici, uno per fare copie di backup, l'altro per ripristinare file memorizzati sull'hard disk. Questi comandi devono essere preferiti al comando COPY, poichè essi tengono in considerazione il fatto che i file dell'hard disk possono essere più grandi della capacità di un singolo dischetto. In particolare, è necessario usare questi comandi quando volete fare una copia di un file per ragioni di sicurezza o quando volete ristrutturare l'hard disk.

ESEMPIO

Il seguente comando, BACKUP, memorizzerà una copia di backup di un file dell'hard disk su un dischetto formattato caricato nel drive A

backup c:filename a:

Il processo di BACKUP inizia così a trasferire il file identificato da "filename" dall'hard disk sul dischetto che si trova nel drive A. La copia di backup avrà lo stesso nome del file originale. Se il file originale è più grande della capacità del dischetto destinatario, l'utility di BACKUP emette un messaggio, nel momento appropriato, che chiede di caricare nel drive A un nuovo dischetto formattato. Questo messaggio viene ripetuto fino a che la procedura di backup non è completata.

Il comando RESTORE trasferisce i file, salvati con il comando BACKUP, sull'hard disk. Per ripristinare un file introducete il seguente comando:

restore a: c:filename

Questo comando trasferirà il file "filename" dal dischetto presente nel drive A sull'hard disk.

Nota: I comandi BACKUP e RESTORE sono molto potenti e flessibili, particolarmente quando vengono usate directory o sub-directory. Per una descrizione completa delle opzioni fornite da questi comandi, vi rimandiamo al manuale "MS-DOS Guida Utente".

INDICE ANALITICO

H

I

L

M

N

INDICE ANALITICO

AVVISO

La Ing. C. Olivetti & C., S.p.A. si riserva il diritto di apportare modifiche al prodotto descritto in questo manuale in qualsiasi momento e senza preavviso.

Il presente manuale è concesso in uso al Cliente alle condizioni specificate nella Licenza d'uso acclusa al Programma cui il manuale si riferisce.

Code 4037180 H (0)
Printed in Italy

olivetti
PERSONAL
COMPUTER

olivetti

Operating Systems and Languages Library

MS GW-BASIC INTERPRETER per MS-DOS

Guida Utente

olivetti
PERSONAL
COMPUTER

olivetti

PREFAZIONE

Questa pubblicazione comprende sia una guida utente sia una parte di consultazione per il linguaggio GW-BASIC (Microsoft Ver-2.01 e 3.11) disponibile sui Personal Computer Olivetti.

SOMMARIO

Il manuale è diviso in due parti.

La parte I, i primi sette capitoli, comprende: l'attivazione, i modi operativi, l'editor di video, la programmazione in generale, l'input/output su disco, la grafica, i sottoprogrammi in linguaggio macchina, il rilevamento di eventi, i processi "child" e la comunicazione asincrona.

La parte II, il Capitolo 8, contiene una descrizione dettagliata di tutti i comandi, le istruzioni e le funzioni disponibili, con i relativi esempi.

RIFERIMENTI

Guida all'Installazione e all'Uso (Code 3986500 T)
MS-DOS Operazioni Fondamentali - Guida all'Uso (Code 4013270 X)
MS-DOS Guida Utente (Code 4001800 C)
MS GW-BASIC Compiler Guida Utente (Code 4022750 C)

DISTRIBUZIONE: Generale (G)

QUARTA EDIZIONE: Marzo 1986

PUBBLICAZIONE EMESSA DA:

Ing. C. Olivetti & C., S.p.A.
Direzione Documentazione
77, Via Jervis - 10015 Ivrea (Italy)

- OLIVETTI è un marchio della Ing. C. Olivetti & C., S.p.A.
- OLITALK è un marchio della Ing. C. Olivetti & C., S.p.A.
- OLITERM è un marchio della Ing. C. Olivetti & C., S.p.A.
- SORTP è un marchio della Ing. C. Olivetti & C., S.p.A.
- ASM-86 è un marchio della Digital Research
- CB-86 è un marchio della Digital Research
- CBASIC-86 è un marchio della Digital Research
- CLEO è un marchio della Phone 1 Inc.
- Concurrent CP/M-86 è un marchio della Digital Research
- Concurrent DOS è un marchio della Digital Research
- CP/M-86 è un marchio della Digital Research
- DDT-86 è un marchio della Digital Research
- Dr. Draw è un marchio della Digital Research
- Dr. Graph è un marchio della Digital Research
- Dr. Logo è un marchio della Digital Research
- ETHERNET è un marchio della Xerox Corp.
- GEM COLLECTION è un marchio della Digital Research
- GEM DESKTOP è un marchio della Digital Research
- GEM DRAW è un marchio della Digital Research
- GEM GRAPH è un marchio della Digital Research
- GEM PAINT è un marchio della Digital Research
- GEM WORDCHART è un marchio della Digital Research
- GEM WRITE è un marchio della Digital Research
- GSS è un marchio registrato della Graphic Software Systems, Inc.
- GSS-CHART è un marchio della Graphic Software Systems, Inc.
- GSS-DRIVERS è un marchio della Graphic Software Systems, Inc.
- GSS-KERNEL SYSTEM è un marchio della Graphic Software Systems, Inc.
- GSS-TOOLKIT è un marchio della Graphic Software Systems, Inc.
- GSX-86 è un marchio della Digital Research
- GW è un marchio della Microsoft Corp.
- IBM è un marchio registrato della International Business Machines Corp.
- MICROSOFT è un marchio registrato della Microsoft Corp.
- MS è un marchio della Microsoft Corp.
- OMNINET è un marchio della Corvus Systems Inc.
- p-System è un marchio della Softech Microsystem, Inc.
- PC-DOS è un marchio della International Business Machines Corp.
- PEACHPAK è un marchio della Peachtree Software International Ltd.
- Personal Basic è un marchio della Digital Research
- SID-86 è un marchio della Digital Research
- TerminAll è un marchio della TOPIC System, Inc.
- TOPIC è un marchio della TOPIC System, Inc.
- UCSD e UCSD Pascal sono marchi registrati della Regent of the University of California
- UNIX è un marchio della Bell Laboratories
- Z80 è un marchio registrato della Zilog Inc.
- Z8000 è un marchio registrato della Zilog Inc.

INDICE

PARTE I

1. PROGRAMMAZIONE IN GW-BASIC

2. OPERAZIONI PRELIMINARI

PARTE II

8. COMANDI, ISTRUZIONI E FUNZIONI

PARTE I

1. PROGRAMMAZIONE IN GW-BASIC

SOMMARIO

Questo capitolo fornisce una introduzione all'interprete GW-BASIC mettendo in evidenza le sue prestazioni principali, i requisiti di sistema, il formato della linea, l'insieme di caratteri e le parole riservate.

INDICE

INTRODUZIONE

Il linguaggio GW-BASIC costituisce la più estesa implementazione del BASIC disponibile per i personal computer. Esso soddisfa i requisiti richiesti dallo standard ANSI ed ha molte prestazioni raramente disponibili su altre versioni di BASIC. Inoltre GW-BASIC permette una gestione sofisticata delle stringhe e la programmazione strutturata, che sono caratteristiche particolarmente adatte allo sviluppo di programmi applicativi. Il linguaggio GW-BASIC offre all'utente ampie funzionalità grafiche, facilità d'uso e prestazioni che rendono un personal computer simile ad un minicomputer.

PRINCIPALI CARATTERISTICHE DEL GW-BASIC

GW-BASIC è un linguaggio di programmazione ad alto livello e di facile uso per risolvere problemi di carattere gestionale e scientifico. Di seguito sono riportate alcune delle sue principali caratteristiche:

- Reindirizzamento di Input Standard (INPUT, LINE INPUT, INPUT$, INKEY$) e di Output Standard (PRINT, PRINT USING).
- Directory con struttura ad albero per l'organizzazione di archivi di programmi e dati e comandi per la gestione di directory (MKDIR/CHDIR/RMDIR).
- Utilizzo di un potente Screen Editor.
- Prestazioni grafiche di alto livello: tre modalità di Risoluzione (Media, Alta e Altissima).
- Richiamo da GW-BASIC di routine in linguaggio Assembler, tramite le istruzioni CALL o CALLS o la funzione USR.
- Gestione degli errori e di particolari tipi di "eventi" tramite l'attivazione di routine definite dall'utente.
- Gestione dell'interfaccia standard di comunicazione RS-232-C.

- Funzioni matematiche (ATN, COS, LOG, EXP, SIN, SQR, TAN) in semplice o doppia precisione.

- Controllo dell'allocazione di memoria di GW-BASIC per routine utente, tramite lo switch /M: del comando GWBASIC.

- Concatenamento con variabili comuni per permettere l'uso di programmi di dimensioni più grandi della memoria disponibile.

- Istruzioni di dichiarazione; i nomi di variabile possono essere elencati in un'istruzione di dichiarazione, specificando esplicitamente se la variabile è di tipo intero in semplice o doppia precisone o stringa.

REQUISITI DI SISTEMA

È possibile eseguire l'interprete GW-BASIC sotto il sistema operativo MS-DOS usando la minima configurazione di sistema. È richiesta almeno una unità disco.

CONVENZIONI SINTATTICHE

1. Le lettere e le parole maiuscole (in grassetto) e i simboli sottoelencati (pure in grassetto) devono essere digitati come indicato nei formati sintattici:

    (
)
    ,
    ;
    :
    "
    =
    /
    \
    #
    $
    -
    >
    <

Nell'istruzione:

WRITE # *filenum*, *list-of-expressions*

WRITE, # e la virgola che segue *filenum* devono essere digitati come specificato.

2. Le lettere e le parole minuscole (in corsivo) rappresentano variabili (o parametri) che l'utente deve specificare.

3. I simboli sottoelencati vengono usati per definire la sintassi di una linea, ma non devono essere digitati:

I	tratto verticale (segno che significa "o"), indica diverse alternative
{ }	parentesi graffe, indicano una scelta
[ ]	parentesi quadre, indicano elementi opzionali
...	simbolo di iterazione
-	lineetta d'unione, unisce parametri formati da più nomi. In alcune istruzioni o comandi (per esempio LIST, LLIST ecc.) la lineetta viene usata come operatore per separare i parametri. In questo caso viene usato il grassetto per distinguere le lineette utilizzate per questo scopo da quelle usate per unire parametri formati da più nomi.

4. Le parentesi graffe raggruppano dati omogenei in alternativa tra loro (divisi da un tratto verticale).
   La rappresentazione:

   { A I B I C }

   indica che si deve scegliere uno degli elementi racchiusi nelle parentesi graffe. Può anche venire usata la rappresentazione:

A

oppure

B

oppure

C

5. Anche le parentesi quadre raggruppano dati tra loro correlati (divisi da un tratto verticale); in questo caso, però, tutto ciò che si trova tra parentesi è opzionale e può essere omesso. La rappresentazione:

   [A I B I C]

   indica che, si può scegliere uno dei dati racchiusi tra parentesi quadre od ometterli tutti.

6. Un'iterazione indica che il dato o il gruppo di dati che la precedono possono venire ripetuti più di una volta in sequenza. La rappresentazione:

   A[, B]...

   indica che A può essere digitato da solo o che può essere seguito da

   ,B

   più di una volta in sequenza.

Nota

È permessa anche la rappresentazione

A [,lista di B]

ed ha lo stesso significato di

A[,B]...

7. I caratteri che compaiono in grassetto negli esempi rappresentano linee di esecuzione immediata o caratteri da introdurre a seguito di richieste effettuate con INPUT.

FORMATO DELLA LINEA

Le linee GW-BASIC possono contenere un massimo di 255 caratteri e hanno il seguente formato:

[*nnnnn*] *istruzione* [*:istruzione*]...[*'commento*] **CR**

Una linea all'interno di un programma GW-BASIC inizia sempre con un numero di linea (un intero senza segno compreso nell'intervallo da 1 a 65529) e termina con **CR**. Una linea di programma viene memorizzata non appena viene digitato **CR**.

Una linea immediata GW-BASIC, cioè una linea che viene eseguita appena impostata, inizia sempre con una lettera, poichè in questo caso deve essere omesso il numero di linea.

Su una linea possono venire inserite più istruzioni GW-BASIC, ma ogni istruzione successiva deve essere separata, da quella che la precede, da due punti (:).

Al termine di una linea GW-BASIC (prima di **CR**) l'utente può inserire una stringa di commento preceduta da un apice ('). Una stringa di commento preceduta dalla parola chiave REM o da un apice può anche essere scritta subito dopo il numero di linea.

È possibile dare maggiore estensione ad una linea logica rispetto ad una fisica tramite line feed (**CTRL CR**); questo comando permette all'utente di continuare a digitare una linea logica sulla linea fisica seguente senza premere **CR**. D'ora in avanti, per motivi di brevità, non verrà specificato **CR** alla fine delle linee GW-BASIC.

Esempi

```
10 FOR K = 1 TO 20
```

è una linea di programma GW-BASIC,

```
100 GOSUB 1000 'vai a SUB1
```

è una linea di programma GW-BASIC con l'aggiunta di un commento finale,

```
1000 'SUB1
```

è una linea di programma GW-BASIC contenente soltanto un commento,

PRINT A$

è un linea immediata GW-BASIC.

NUMERI DI LINEA

Ogni linea di programma GW-BASIC inizia con un numero di linea. I numeri di linea indicano l'ordine in cui le linee di programma sono memorizzate. I numeri di linea vengono anche usati come riferimento nelle istruzioni di salto e nelle operazioni di editing e devono essere compresi tra 0 e 65529. Per i comandi EDIT, LIST, AUTO e DELETE può essere usato un punto (.) come riferimento alla linea corrente.

SET DI CARATTERI

GW-BASIC riconosce i seguenti caratteri:

- Caratteri alfabetici (lettere dell'alfabeto, maiuscole e minuscole)
- Caratteri numerici (cifre da 0 a 9)
- Caratteri speciali

Caratteri Speciali

La lista dei caratteri speciali GW-BASIC viene riassunta nella tabella sottoindicata:

CARATTERE	SIGNIFICATO
	Spazio
=	Segno di uguale o simbolo di assegnazione
+	Segno più
—	Segno meno
*	Asterisco o simbolo di moltiplicazione
/	Barra o simbolo di divisione
^	Freccia in alto o simbolo di esponente
(	Parentesi sinistra
)	Parentesi destra
%	Segno di percentuale o carattere di dichiarazione di dato di tipo intero
#	Segno di numero o dichiarazione di dato in doppia precisione
$	Segno di dollaro o carattere di dichiarazione di dato di tipo stringa
!	Punto esclamativo o carattere di dichiarazione di dato in semplice precisione
[	Parentesi quadra sinistra
]	Parentesi quadra destra
,	Virgola
.	Punto o punto decimale
'	Apice (apostrofo)
"	Doppio apice (delimitatore di stringa)
;	Punto e virgola
:	Due punti
&	Congiunzione commerciale
?	Punto interrogativo (abbreviazione di PRINT)
<	Minore di
>	Maggiore di
\	Barra rovescia o simbolo di divisione tra interi
@	Chiocciola
_	Segno di sottolineatura

PAROLE RISERVATE

GW-BASIC comprende istruzioni, comandi, nomi di funzioni e di operatori che vengono considerati parole riservate e che non possono quindi essere usati come nomi di variabili. Quello che segue è l'elenco completo delle parole riservate GW-BASIC.

ABS
AND
ASC
ATN
AUTO
BEEP
BLOAD
BSAVE
CALL
CALLS
CDBL
CHAIN
CHDIR
CHR$
CINT
CIRCLE
CLEAR
CLOSE
CLS
COLOR
COM
COMMON
CONT
COS
CSNG
CSRLIN
CVD
CVI
CVS
DATA
DATE$
DEF
DEFDBL
DEFINT
DEFSNG
DEFSTR
DELETE
DIM
DRAW
EDIT
ELSE
ELSE
END
ENVIRON
ENVIRON$
EOF
EQV
ERASE
ERDEV
ERDEV$
ERL
ERR
ERROR
EXP
FIELD
FILES
FIX
FNxxxxxxxx
FOR
FRE
GET
GOSUB
GOTO
HEX$
IF
INKEY$
INP
INPUT
INPUT#
INPUT$
INSTR
INT
IOCTL
IOCTL$
KEY
KILL
LEFT$
LEN
LET
LINE
LIST
LLIST
LOAD
LOC
LOCATE
LOCK
LOF
LOG
LPOS
LPRINT
LSET
MERGE
MID$
MKDIR
MKD$
MKI$
MKS$
MOD
NAME
NEW
NEXT
NOT
OCT$
OFF
ON
OPEN
OPTION
OR
OUT
PAINT
PEEK

PALY
PMAP
POINT
POKE
POS
PRESET
PRINT
PRINT#
PSET
PUT
RANDOMIZE
READ
REM
RENUM
RESET
RESTORE
RESUME
RETURN
RIGHT$
RMDIR
RND
RSET
RUN
SAVE
SCREEN
SGN
SHELL
SIN
SOUND
SPACE$
SPC
SQR
STEP
STOP
STR$
STRING$
SWAP
SYSTEM
TAB
TAN
THEN
TIMER
TIME$
TO
TROFF
TRON
UNLOCK
USING
USR
VAL
VARPTR
VARPTR$
VIEW
WAIT
WEND
WHILE
WIDTH
WINDOW
WRITE#
WRITE
XOR

2. OPERAZIONI PRELIMINARI

SOMMARIO

Questo capitolo spiega come entrare ed uscire dall'ambiente GW-BASIC, i modi di operare, l'uso della tastiera e l'editor di video. Inoltre insegna come impostare, listare, registrare, caricare, modificare ed eseguire un programma GW-BASIC.

INDICE

PROCEDURA DI INIZIALIZZAZIONE

Prima di attivare l'ambiente GW-BASIC è necessario installare il sistema operativo MS-DOS. Dopo l'installazione di MS-DOS e la visualizzazione del prompt del sistema:

```
A>
```

l'utente deve impostare il comando:

GWBASIC

per trasferire in memoria GW-BASIC, dal diskette inserito nel drive A.

Terminato il caricamento di GW-BASIC l'utente può effettuare una delle seguenti operazioni:

- Inserire un diskette contenente i programmi GW-BASIC ed eseguirne uno; per esempio:

  run "b:*myfile*"

  dove *myfile* è il programma GW-BASIC che l'utente vuole eseguire e che è memorizzato sul diskette inserito nel drive B.

- Inserire il programma GW-BASIC o linee di programma, per esempio:

  10 K = 10
  20 FOR J = 1 TO 2

  Queste sono linee di programma che, dopo essere state digitate, vengono conservate in memoria. Per eseguire un programma l'utente deve impostare il comando RUN.

Osservazioni

Il comando GWBASIC può essere digitato con diverse opzioni che ottimizzano l'occupazione della memoria e permettono il reindirizzamento di Input o Output Standard, ecc...

COME USCIRE DALL'AMBIENTE GW-BASIC

Per uscire dall'ambiente GW-BASIC e ritornare in ambiente MS-DOS l'utente deve impostare:

SYSTEM

Questo comando chiude tutti i file dati prima di ritornare in ambiente MS-DOS. Il programma GW-BASIC viene così cancellato dalla memoria centrale mentre MS-DOS rimane residente.

Quando viene di nuovo visualizzato il prompt di sistema (A>) l'utente può impostare un altro comando MS-DOS.

MODI OPERATIVI

L'Interprete GW-BASIC può essere usato in due modi: "modo immediato" o "modo programma".

Nel modo immediato (o "diretto"), le istruzioni e i comandi vengono eseguiti immediatamente dopo essere stati digitati e non sono preceduti da un numero di linea. Dopo ogni istruzione diretta, seguita da **CR**, viene visualizzato "Ok".

L'utente può visualizzare immediatamente i risultati di operazioni aritmetiche e logiche e memorizzarli per usarli in seguito, ma le istruzioni non vengono conservate dopo la loro esecuzione. Questo modo è utile per il debugging o per calcoli veloci che non richiedono un programma completo.

Esempio

```
Ok
PRINT 45+3
 48
Ok
```

Il modo programma (o "indiretto") viene usato per inserire programmi. Le linee di programma, precedute da un numero di linea, vengono memorizzate come parte di un programma. Il programma memorizzato può essere eseguito tramite il comando RUN.

Esempio

```
Ok
10 PRINT 45+3
RUN
 48
Ok
```

LA TASTIERA

Sono disponibili due diversi tipi di tastiera per i PC Olivetti: la TASTIERA 1 e la TASTIERA 2. Entrambe hanno diverse versioni nazionali.

Vedere "GUIDA ALL'INSTALLAZIONE E ALL'USO" (3986500T) per una descrizione completa delle tastiere disponibili.

TASTI FUNZIONE

Sono disponibili 10 tasti funzione sulla tastiera 1 e 18 sulla tastiera 2.

Questi tasti funzione possono essere adattati alle esigenze dell'utente tramite le istruzioni KEY e ON KEY (*n*) GOSUB.

Per assegnare a un tasto funzione un comando o una sequenza di caratteri specifici, diversi dai comandi standard già assegnati, è possibile usare l'istruzione KEY. Per interrompere il programma tramite un tasto funzione specificato è possibile usare l'istruzione ON KEY (*n*) GOSUB.

Per ulteriori dettagli occorre consultare il Capitolo 8.

TASTIERA ALFANUMERICA

La tastiera alfanumerica standard, collocata al centro, viene usata per introdurre lettere, numeri, caratteri speciali e caratteri di controllo.

Tasti Shift

Se l'utente vuole impostare un carattere maiuscolo o il simbolo superiore dei tasti contenenti due simboli, deve premere contemporaneamente uno dei due tasti ↑ (sulla tastiera 1) o uno dei due tasti **SHIFT** (sulla tastiera 2) e il tasto corrispondente. D'ora in avanti, per convenzione, si farà sempre riferimento ai tasti ↑ come tasti **SHIFT**.

Tasto di Chiusura Impostazione

Il tasto di chiusura impostazione viene identificato dal simbolo ↵. Per convenzione ci riferiremo a questo tasto come al tasto **CR**.

L'utente deve premere **CR** per completare una linea GW-BASIC e inviarla al sistema per l'elaborazione.

Fissa Maiuscolo per le Lettere

L'utente può abilitare o disabilitare il fissa maiuscolo per le lettere (A-Z) premendo **CAPS LOCK**.

Il tasto **CAPS LOCK** è simile al tasto fissa maiuscole di una macchina per scrivere, ma permette solo l'introduzione di lettere maiuscole e non dei simboli superiori dei tasti con due simboli.

Tasto di Ritorno di un Passo

Il tasto di ritorno di un passo ← (sulla tastiera 1) oppure **BS** (sulla tastiera 2) sposta il cursore di una posizione a sinistra cancellando l'ultimo carattere digitato.

Per spostare il cursore a sinistra, senza cancellare nessun carattere, si usi il tasto Cursor Left (cursore a sinistra).

Caratteri di Controllo

I caratteri di controllo vengono impostati premendo contemporamente il tasto **CTRL** o il tasto **ALT** e un altro tasto. GW-BASIC riconosce i seguenti caratteri di controllo:

CARATTERE DI CONTROLLO	SIGNIFICATO
CTRL BREAK	CTRL BREAK può essere usato per diversi scopi: 1. Per interrompere l'esecuzione di un programma alla successiva istruzione GW-BASIC e ritornare in Stato Comandi GW-BASIC. 2. Per interrompere la numerazione automatica delle linee durante la digitazione di un programma.

CARATTERE DI CONTROLLO	SIGNIFICATO
	3. Per ritornare in Stato Comandi senza conservare nessuno dei cambiamenti fatti alla linea corrente.
CTRL G	Attiva il segnalatore acustico.
CTRL NUM LOCK (sulla tastiera 1) **CTRL FUNCT LOCK** (sulla tastiera 2)	Produce una "pausa" nel sistema interrompendo temporaneamente la stampa o il listing del programma. La pausa continua finché non viene premuto un qualsiasi altro tasto (eccetto **SHIFT**, **CTRL** o **ALT**).
CTRL T	Per video da 40 caratteri, attiva e disattiva la visualizzazione dei 10 tasti funzione (sulla 25ª linea del video). Per video da 80 caratteri, commuta su ON e OFF la funzione di visualizzazione.
ALT CTRL DEL	Premendo contemporaneamente i tasti **CTRL** e **ALT** e poi il tasto **DEL** viene eseguito un reset di sistema.
CTRL PRTSC (sulla tastiera 1) **CTRL SCR PRT** (sulla tastiera 2)	Il testo visualizzato viene anche stampato; è possibile interrompere la stampa ripetendo la sequenza dei tasti. Se vengono premuti contemporaneamente **PRTSC** e **SHIFT** sulla tastiera 1, oppure se viene premuto **SCR PRT** sulla tastiera 2, MS-DOS stampa una copia del contenuto del video.

▶

CARATTERE DI CONTROLLO	SIGNIFICATO
	Per stampare sia il testo sia la parte grafica è necessario usare il comando di MS-DOS GRAPHICS prima di entrare in GW-BASIC.
CTRL L	Produce un carattere di avanzamento carta ed ha la stessa funzione dell'istruzione CLS, cioè cancella tutti i caratteri dal video o dalla finestra di grafica corrente (se ne è stata definita una).
CTRL Z	Determina una condizione di "end of file". Vedere l'istruzione OPEN COM.

Altri caratteri di controllo sono descritti nella sottosezione di questo capitolo intitolata "TASTI SPECIALI DI SCREEN EDITOR".

Impostazione Diretta di Parole Chiave GW-BASIC

Una parola chiave GW-BASIC viene impostata premendo contemporaneamente il tasto **ALT** e uno dei tasti alfabetici da A a Z. L'associazione di ogni lettera dell'alfâbeto con la relativa parola chiave viene riportata nell'elenco seguente.

A	- AUTO	N	- NEXT
B	- BLOAD	O	- OPEN
C	- COLOR	P	- PRINT
D	- DELETE	Q	- ****
E	- ELSE	R	- RUN
F	- FOR	S	- SCREEN
G	- GOTO	T	- THEN
H	- HEX$	U	- USING
I	- INPUT	V	- VAL
J	- ****	W	- WIDTH
K	- KEY	X	- XOR
L	- LOCATE	Y	- ****
M	- MERGE	Z	- ****

**** = tasti inutilizzati

TASTIERA NUMERICA

Un gruppo di 15 tasti sul lato destro della tastiera 1 e di 27 tasti sullo stesso lato della tastiera 2 è disposto in modo molto simile a quello di una tastiera di un normale calcolatore ed è chiamato "tastiera numerica". Comprende non solo i tasti con i numeri da 0 a 9, il punto decimale, i tasti di addizione (+) e sottrazione (-), ma anche i tasti di movimento del cursore, **PG UP**, **PG DN**, **HOME**, **NUM LOCK** (sulla tastiera 1), **FUNCT LOCK** (sulla tastiera 2), **SCROLL LOCK**, **BREAK**, **END**, **INS**, **DEL**, ecc.

Si noti che alcuni tasti come **SCROLL LOCK**, **PG UP**, **PG DN**, ecc., non vengono usati da GW-BASIC, ma è possibile assegnare loro dei significati all'interno di un programma GW-BASIC.

Stato Fissa Numeri (Tastiera 1)

Per predisporre la tastiera numerica in modo maiuscolo si prema il tasto **NUM LOCK** (sulla tastiera 1). In questo modo sono disponibili i numeri da 0 a 9 e il punto decimale. (Per digitare i corrispondenti tasti minuscoli è necessario premere uno dei due tasti **SHIFT**).

Premendo **NUM LOCK** un'altra volta la tastiera ritorna in modo minuscolo.

Stato Fissa Funzioni (Tastiera 2)

Per predisporre la tastiera numerica in modo minuscolo si prema il tasto **FUNCT LOCK**. In questo modo sono disponibili le funzioni (come **HOME, INS, DEL, END, PGUP, PGDN**). Per digitare i corrispondenti tasti maiuscoli è necessario premere uno dei due tasti **SHIFT**. Premendo **FUNCT LOCK** un'altra volta la tastiera ritorna in modo maiuscolo.

SCREEN EDITOR GW-BASIC

Tutti i testi impostati mentre GW-BASIC si trova in Stato Comandi vengono elaborati dall'Editor GW-BASIC. Questo è un "Editor di Linea di Video" che permette all'utente di modificare una linea in qualsiasi punto del video (una sola linea alla volta). Le modifiche vengono registrate solo quando l'utente preme **CR** sulla linea modificata.

TASTI SPECIALI DI SCREEN EDITOR

L'Editor GW-BASIC riconosce 9 tasti della tastiera numerica, il tasto di Ritorno di un Passo e il tasto **CTRL** per muovere il cursore, inserire o cancellare i caratteri.

La seguente tabella riporta i tasti e le loro funzioni.

TASTO	FUNZIONE
HOME	HOME Sposta il cursore verso l'angolo del video in alto a sinistra.
CTRL HOME	CLEAR SCREEN Cancella tutti i caratteri presenti sul video e sposta il cursore alla posizione "Home".
↑	CURSOR UP (CURSORE IN ALTO) Sposta il cursore di una linea verso l'alto.

▶

TASTO	FUNZIONE
↓	CURSOR DOWN (CURSORE IN BASSO) Sposta il cursore di una linea verso il basso.
←	CURSOR LEFT (CURSORE A SINISTRA) Sposta il cursore di una posizione verso sinistra. Se il cursore viene mosso oltre il limite sinistro del video, ricompare sul lato destro del video sulla linea superiore.
→	CURSOR RIGHT (CURSORE A DESTRA) Sposta il cursore di una posizione verso destra. Se il cursore viene mosso oltre il limite destro del video, ricompare sul lato sinistro del video sulla linea sottostante.
CTRL →	NEXT WORD (PAROLA SEGUENTE) Sposta il cursore verso la parola seguente, cioè verso il primo carattere, alla destra del cursore, (compreso nei caratteri [A...Z] oppure [a...z] o nei numeri [0...9]) che segue uno spazio o un carattere speciale. Per esempio nella linea seguente: 30 IF L< =0 THEN 20 Il cursore è sotto la lettera L. Premendo **CTRL →**, il cursore si posta sotto alla parola seguente, in qualsiasi caso 0: 30 IF L < =0 THEN 20 Premendo nuovamente **CTRL →** il cursore si

TASTO	FUNZIONE
	sposta verso la parola seguente che è THEN: 30 IF L< =0 THEN 20
CTRL ←	PREVIOUS WORD (PAROLA PRECEDENTE) Sposta il cursore all'inizio della parola precedente, cioè al primo carattere alla sinistra del cursore (compreso tra [A...Z] o [a...z] o [0...9]) preceduto da uno spazio o da un carattere speciale. Per esempio: 30 IF L< =0 THEN 20 Il cursore si trova sotto la lettera T. Premendo **CTRL ←**, il cursore si sposta sotto 0. Premendo **CTRL ←** il cursore si sposta sotto L.
END	END (FINE) APPEND (AGGIUNTA) Sposta il cursore dalla posizione corrente alla fine della linea logica. Caratteri successivi digitati da tastiera vengono aggiunti alla fine della linea.
CTRL END	ERASE TO END OF LINE (CANCELLAZIONE FINO ALLA FINE DELLA LINEA) Cancella dalla posizione corrente del cursore fino alla fine della linea logica, cioè fino al tasto di chiusura impostazione **CR**.
INS	SWITCH INSERT/OVERWRITE MODE (COMMUTATORE INSERIMENTO / SCRITTURA). Abilita o disabilita lo Stato Inserimento. L'abilitazione di uno dei due stati comporta la disabili-

TASTO	FUNZIONE
	tazione dell'altro. Lo Stato Inserimento viene disabilitato anche premendo uno qualsiasi dei tasti per lo spostamento del cursore o **CR**. In Stato Inserimento il carattere che si trova immediatamente sopra il cursore e quelli che lo seguono si spostano verso destra, non appena vengono inseriti nuovi caratteri nella posizione del cursore. I caratteri che scompaiono sul lato destro del video riappaiono all'inizio della linea successiva. In Stato Inserimento il cursore è rappresentato da un rettangolo che lampeggia, (nel Modo Testo) o da un triangolo che lampeggia alla sinistra di ogni carattere (nei modi di grafica). Lo Stato Scrittura è indicato da un cursore diverso, e cioè da un segno di sottolineatura che lampeggia più lentamente. In Stato Scrittura i caratteri digitati sostituiscono i caratteri esistenti sulla linea.
→|	TAB (TABULAZIONE) Una volta disabilitato lo Stato Inserimento, premendo →| il cursore viene spostato fino a raggiungere il punto di tabulazione seguente. I punti di tabulazione sono fissati a intervalli di 8 caratteri a partire dalla posizione 1. Per esempio nella linea: 20 INPUT "Length"; L Premendo →| il cursore si sposta alla diciassettesima posizione in questo modo:

TASTO	FUNZIONE
	20 INPUT "Length"; L In Stato Inserimento, premendo →| vengono inseriti degli spazi dalla posizione corrente del cursore fino al punto di tabulazione successivo. I caratteri che scompaiono alla destra del video, riappaiono all'inizio della linea seguente. Per esempio nella linea: 20 INPUT "Length"; L Vengono inseriti degli spazi fino alla diciassettesima posizione premendo **INS** e poi →|. 20 INPUT " Length"; L
DEL	DELETE CHARACTER (CANCELLAZIONE DI CARATTERI) Cancella il carattere nella posizione corrente del cursore. Tutti i caratteri che seguono il carattere cancellato si spostano di una posizione a sinistra. Se una linea logica si estende oltre una linea fisica, i caratteri che si trovano sulle linee seguenti si spostano a sinistra di una posizione e il carattere che si trova sulla prima colonna di ogni linea successiva si sposta in alto alla fine della linea precedente.
← sulla Tastiera 1 o **BS** sulla Tastiera 2	BACKSPACE (RITORNO INDIETRO DI UNA POSIZIONE) Cancella l'ultimo carattere digitato oppure quello alla sinistra del cursore. Tutti i caratteri alla destra del carattere cancellato e i caratteri delle linee all'interno della linea logica corrente si spostano in alto come nella funzione precedente (**DEL**).

TASTO	FUNZIONE
CTRL CR	LINE FEED (RITORNO A CAPO) Fa iniziare il testo seguente automaticamente sulla successiva linea del video.
ESC	DELETE LINE (CANCELLAZIONE DI LINEE) L'intera linea logica contenente il cursore viene cancellata e non viene elaborata. Se è una linea di programma non viene cancellata dal programma in memoria.
CTRL BREAK	BREAK (INTERRUZIONE) Ritorna in Stato Comandi senza salvare nessuna delle modifiche fatte alla linea corrente. Al contrario della funzione ESC, non cancella la linea dal video.

COME CORREGGERE LA LINEA CORRENTE

Tutti i testi impostati in Stato Comandi GW-BASIC vengono elaborati dallo Screen Editor. L'utente può quindi usare tutti i Tasti Speciali di Screen Editor descritti precedentemente.

GW-BASIC rimane in Stato Comandi dopo il prompt Ok finchè non riceve il comando RUN.

Correzione di Caratteri

Se viene commesso un errore in fase di digitazione di una linea, si può procedere come segue:

PASSO	AZIONE
1	Individuare l'errore. Supponiamo per esempio che sia stato digitato: **RUN "K,PROGR_** invece di: **RUN "A:PROGR_**
2	Usare il tasto Cursor Left, o altri tasti di movimento del cursore, per spostare il cursore nella posizione appropriata: **RUN "K,PROGR**
3	Digitare i caratteri corretti su quelli errati: **RUN "A:PROGR**
4	Spostare il cursore fino alla fine della linea usando il tasto Cursor Right oppure il tasto **END**: **RUN "A:PROGR_**
5	Continuare a digitare se la linea non è terminata: **RUN "A:PROGRAM1"_**
6	Premere **CR** per passare la linea a GW-BASIC. In questo caso il programma specificato viene caricato dal diskette inserito nel drive A ed eseguito.

Inserimento di Caratteri

Se l'utente omette dei caratteri nella linea che sta impostando, può procedere in questo modo:

PASSO	AZIONE
1	Individuare l'errore. Supponiamo che sia stato digitato: **10 FO K=1 TO_** invece di: **10 FOR K=1 TO_**
2	Usare il tasto Cursor Left o altri tasti di movimento del cursore per spostare il cursore nella posizione appropriata: **10 FO_K=1 TO**
3	Premere **INS** e digitare la lettera R: **10 FOR_K=1 TO** Si noti che, entrando in Stato Inserimento, il cursore assume altezza doppia.
4	Premere nuovamente **INS** per ritornare in Stato Scrittura e poi il tasto Cursor Right o **END** per spostare il cursore fino alla fine della linea: **10 FOR K=1 TO_**

Cancellazione di Caratteri

Se l'utente introduce un carattere in piú sulla linea che sta impostando, può procedere in questo modo:

PASSO	AZIONE
1	Individuare l'errore. Supponiamo che sia stato digitato: **GOTTO_** invece di: **GOTO_**
2	Per cancellare la T superflua, premere il tasto Cursor Left, o altri tasti di movimento del cursore, per spostare il cursore nella posizione appropriata: **GOTTO**
3	Premere **DEL** **GOTO**
4	Spostare il cursore usando il tasto Cursor Right. **GOTO_**
5	Continuare a digitare: **GOTO 1000_**

Come Cancellare Parte di una Linea

Per cancellare una linea dalla posizione corrente del cursore in poi si deve premere **CTRL END**.

Come Cancellare un'Intera Linea

Per cancellare la linea che l'utente sta impostando si deve premere **ESC** in qualsiasi punto della linea. Non è necessario premere **CR**.

COME MODIFICARE LE LINEE DI PROGRAMMA

Ogni linea di testo che inizia con un numero (da 0 a 65529) è considerata una linea di programma. Se si vuole modificare un programma precedentemente introdotto occorre procedere come segue:

SE l'utente vuole...	ALLORA...
aggiungere una nuova linea al programma	deve impostare un numero di linea valido seguito da almeno un carattere, che non sia uno spazio, e da **CR**.
sostituire una linea di programma	deve impostare un numero di linea uguale ad uno già esistente, seguito dal contenuto della nuova linea.
cancellare una linea	deve impostare il numero della linea da cancellare seguito da **CR**. Se l'utente cerca di cancellare una linea non esistente viene visualizzato il messaggio di errore "Undefined line number". Nota: Non si deve usare **ESC** per cancellare linee di ▶

SE l'utente vuole...	ALLORA...
	programma poichè cancella la linea solo su video, mentre rimane nel programma.
cancellare un gruppo di linee	deve impostare il comando DELETE indicando il numero di linee da cancellare.
cancellare il programma residente in memoria	deve impostare il comando NEW.
modificare una linea di programma già presente sul video	deve spostare il cursore alla posizione appropriata (tramite i tasti di movimento del cursore); modificare la linea usando una delle tecniche illustrate precedentemente per modificare, cancellare o inserire caratteri in una linea; premere quindi **CR** per passare la linea modificata a GW-BASIC.
modificare una linea di programma non presente sul video	deve usare il comando EDIT, per visualizzare la linea, o il comando LIST, per visualizzare un gruppo di linee che comprende la linea da modificare, spostare il cursore alla posizione appropriata, modificare la linea e premere **CR**. Nota: una linea può solo essere editata finchè è presente sul video. Quando una linea immediata è stata inviata al sistema, premendo **CR**, non è più possibile editarla; questo non avviene con le linee di programma, che possono sempre essere richiamate sul video per essere modificate.

Osservazioni

Le modifiche fatte a una linea vengono memorizzate quando viene premuto **CR**, mentre il cursore si trova in un qualsiasi punto della stessa linea. La funzione **CR** permette di introdurre modifiche su una linea logica, per un massimo di 255 caratteri, indipendentemente dal numero di linee fisiche e dal punto in cui si trova il cursore nella linea logica.

Si noti che le modifiche fatte tramite lo Screen Editor cambiano il programma solo in memoria. Per salvare queste modifiche su un file su disco bisogna usare il comando SAVE.

USO DEL SISTEMA COME CALCOLATORE

Il Personal Computer può essere usato come un calcolatore per calcoli veloci e per operazioni di debugging.

Quando il sistema è in ambiente GW-BASIC e il prompt Ok compare su video l'utente può impostare PRINT (o semplicemente ?), seguito da una qualsiasi espressione e da **CR**. L'espressione viene calcolata e il risultato viene visualizzato.

L'utente può anche impostare LET, seguito da un qualsiasi nome di variabile, l'operatore di assegnazione (=), una qualsiasi espressione e **CR**. L'espressione viene calcolata e il risultato assegnato alla variabile specificata, che può essere utilizzata per rappresentare quel determinato valore in calcoli successivi. La parola chiave LET è opzionale, l'utente può iniziare la linea usando semplicemente il nome della variabile.

La seguente tabella riporta alcuni esempi.

VISUALIZZAZIONE	COMMENTO
PRINT 3 3 Ok	Viene visualizzata la costante 3.

VISUALIZZAZIONE	COMMENTO
PRINT 2+3 5 Ok	Viene calcolata l'espressione 2 + 3 e viene visualizzato il risultato (5).
LET A=15.21 Ok	La costante 15.21 viene assegnata alla variabile A. L'utente può utilizzare la variabile A in calcoli successivi per rappresentare questo valore.
?A-1 14.21 Ok	Viene calcolata l'espressione A-1 e viene visualizzato il risultato (14.21). Nota: ? è equivalente a PRINT.
B=2.3 Ok	La costante 2.3 viene assegnata alla variabile B. La parola chiave LET è opzionale, l'utente può iniziare la linea con un nome di variabile.
?A*B 34.983 Ok	Viene calcolata l'espressione A*B e viene visualizzato il risultato (34.983).
?A*B-40 -5.017002 Ok	Viene calcolata l'espressione A*B-40 e viene visualizzato il risultato (-5.017002). Nota: se un valore è negativo viene visualizzato il segno meno, se un valore è positivo non viene visualizzato nessun segno.

COME DIGITARE UN PROGRAMMA

Un programma GW-BASIC si compone di una serie di istruzioni. Un'istruzione è una entità elementare di programma che ordina al computer di svolgere una specifica operazione.

È possibile impostare linee con una o più istruzioni. In questo ultimo caso ogni istruzione viene separata dall'altra da due punti (:).

In un programma GW-BASIC ogni linea inizia con un numero intero maggiore o uguale a 0 e minore o uguale a 65529 e termina quando l'utente preme **CR**.

Una linea GW-BASIC può contenere un massimo di 255 caratteri, compreso **CR**. I caratteri eccedenti vengono troncati quando viene digitato **CR**.

Quando il sistema è sotto il controllo di GW-BASIC e il prompt Ok compare su video, l'utente può inserire un programma. Si digiti dapprima:

```
NEW
```

per cancellare la memoria.

Si imposti poi:

```
10 REM RECT1
20 INPUT "Length";L
30 IF L< =0 THEN 20
40 INPUT "Width";W
50 IF W< =0 THEN 40
60 LET AREA=L*W
70 PRINT "Area= ";AREA;" L= ";L;" W= ";W
80 GOTO 20
90 END
```

Per convenzione la numerazione delle linee è a intervalli di 10. Questo permette all'utente di inserire nuove istruzioni fra le linee già esistenti.

Queste istruzioni formano un programma completo che permette di risolvere un problema molto semplice: trovare l'area di un rettangolo introducendo da tastiera i valori della base e della

altezza. Questo esempio è stato scelto sia per la sua semplicità, sia per illustrare varie prestazioni del linguaggio GW-BASIC. Esistono anche altre soluzioni più concise.

NUMERAZIONE AUTOMATICA DELLE LINEE

L'utente può usare il comando AUTO per generare automaticamente un numero di linea ad ogni pressione del tasto **CR**. Per uscire dal modo AUTO occorre premere **CTRL BREAK**.

COME LISTARE UN PROGRAMMA

Un programma caricato in memoria può essere listato su video mediante il comando LIST e su stampante mediante il comando LLIST.

I comandi LIST e LLIST listano il programma convertendo tutte le parole chiave, i nomi di variabile e i nomi di funzione in lettere maiuscole e convertendo ogni punto interrogativo (?), usato al posto della parola chiave PRINT, in PRINT. Inoltre le istruzioni vengono ordinate in una sequenza crescente di numeri di linea, indipendentemente dall'ordine in cui sono state impostate.

Per listare su video il programma dell'esempio si digiti:

LIST

Viene visualizzato:

```
LIST
10 REM RECT1
20 INPUT "Length";L
30 IF L< =0 THEN 20
40 INPUT "Width";W
50 IF W< =0 THEN 40
60 LET AREA=L*W
70 PRINT "Area=";AREA;" L=";L;" W=";W
80 GOTO 20
90 END
Ok
```

Alla fine del listing il sistema entra in Stato Comandi e visualizza il prompt Ok; il programma può ora essere editato nel modo desiderato.

COME REGISTRARE UN PROGRAMMA

Un programma rimane in memoria finchè il computer è in funzione. Appena il computer viene spento o viene effettuato un reset del sistema, il programma viene perso. Se l'utente vuole memorizzare il programma che ha appena digitato per usarlo in futuro deve impostare un comando SAVE per registrare il programma su disco.

È necessario registrare il programma residente in memoria nei seguenti casi:

SE l'utente vuole...	ALLORA...
spegnere la macchina (o effettuare il reset del sistema)	deve registrare il programma residente in memoria (a meno che non ne esista già una copia su disco). Per esempio: **SAVE "A:RECT1"**
introdurre un altro programma da tastiera	deve registrare il programma residente in memoria (a meno che non ne esista già una copia su disco).
trasferire un altro programma da disco in memoria (tramite un comando LOAD o RUN)	deve registrare il programma residente in memoria (a meno che non esista già una copia su disco).
ritornare in MS-DOS (tramite il comando SYSTEM)	deve registrare il programma residente in memoria (a meno che non ne esista già una copia su disco).

SE l'utente vuole...	ALLORA...
sostituire la versione precedente del programma	deve registrare il programma residente in memoria sullo stesso disco e con lo stesso nome di file della versione precedente.
registrare il programma residente in formato ASCII (sorgente)	deve specificare l'opzione A nel comando SAVE. Per esempio: **SAVE "A:RECT1", A**
proteggere il programma residente in memoria contro ogni tentativo di modificarlo, listarlo o registrarlo di nuovo	deve specificare l'opzione P nel comando SAVE. Per esempio: **SAVE "A:RECT1",P**

COME CARICARE UN PROGRAMMA

Se il programma da trasferire in memoria principale risiede su disco, l'utente deve impostare il comando LOAD. Il comando LOAD cancella tutte le variabili e le linee di programma residenti in memoria, per cui, prima di digitare il comando LOAD, l'utente deve registrare su disco il programma residente in memoria se intende usarlo ancora in futuro.

Per caricare da disco un file programma è necessario specificare il drive prima del nome del file, se il file non risiede sul drive di default. Per esempio; se il programma ROOT1 risiede sul diskette inserito nel drive B:

LOAD "B:ROOT1"

Se viene specificata l'opzione R, tutti i file dati rimangono aperti e il nuovo programma viene eseguito dopo essere stato trasferito in memoria. Per esempio:

LOAD "B:ROOT1",R

Se invece non viene specificata l'opzione R, il comando LOAD chiude tutti i file dati.

COME ESEGUIRE UN PROGRAMMA

Quando un programma è residente in memoria può essere eseguito. Per richiedere l'esecuzione di un programma l'utente deve digitare il comando RUN (o il comando LOAD con l'opzione R).

Il comando RUN esegue il programma residente in memoria, o trasferisce in memoria un programma da disco e lo esegue (se l'utente imposta un identificatore di file dopo RUN). Per esempio:

RUN "B:RECT1"

Si noti che un identificatore di file è un'espressione stringa o, in particolare, una costante stringa; se è una costante stringa, come nell'esempio precedente, deve essere racchiuso da doppi apici (").

Se l'utente specifica l'opzione R tutti i file dati aperti rimangono aperti, perciò l'utente può usare ancora questi file nel nuovo programma senza doverli aprire un'altra volta.

Prima di impostare RUN *filespec* (o RUN *filespec*, R) l'utente deve registrare il programma residente in memoria (se non ne esiste già una copia).

Le istruzioni GW-BASIC vengono eseguite nell'ordine indicato dai numeri di linea, se un'istruzione di controllo (GOTO, ON...GOTO, IF...GOTO...ELSE, IF...THEN...ELSE, FOR/NEXT, WHILE/WEND) o un'istruzione che richiama un sottoprogramma (GOSUB, ON...GOSUB) non specifica altrimenti.

ESEMPIO DI ESECUZIONE DI UN PROGRAMMA

Si vuole eseguire ora il programma descritto nei paragrafi precedenti, già in memoria, introdotto da tastiera o caricato da disco tramite il comando LOAD.

Si digiti:

LIST

per controllare che questo programma sia nella memoria principale. Alla fine del listing, quando il prompt Ok compare sul video, si digiti:

RUN

La sequenza visualizzata è:

```
LIST
10 REM RECT1
20 INPUT "Length";L
30 IF L< =O THEN 20
40 INPUT "Width";W
50 IF W< =O THEN 40
60 LET AREA = L*W
70 PRINT "Area = ";AREA;" L = ";L;" W = ";W
80 GOTO 20
90 END
Ok
RUN
Length? 3.5
Width? 4.2
Area= 14.7   L=3.5   W=4.2
Length? -7.3
Length? 7.3
Width? 1.3Q
?Redo from start
Width? 1.32
Area= 9.636   L=7.3   W=1.32
Length? CTRL BREAK
Break in 20
Ok
```

Il sistema inizia con l'esecuzione sequenziale delle istruzioni. L'istruzione 10 non viene eseguita perchè è una REM (ark), cioè un commento. In questo caso l'esecuzione inizia dall'istruzione 20.

Quando incontra un'istruzione INPUT (istruzioni 20 e 40) l'esecuzione del programma viene sospesa e il sistema visualizza un messaggio che indica all'utente di introdurre un valore. L'utente può introdurre per esempio 3.5 per la base e 4.2 per l'altezza.

L'istruzione 60 calcola il valore dell'AREA. L'istruzione 70 visualizza il valore dell'area di L e di W. L'istruzione 80 restituisce il controllo all'istruzione 20.

Se viene digitato un valore negativo (per esempio -7.3) per L viene eseguita nuovamente l'istruzione 20, perchè l'istruzione 30 restituisce il controllo all'istruzione 20 se L ha un valore minore o uguale a zero.

Se viene digitato un valore negativo per W l'istruzione 40 viene eseguita un'altra volta, l'istruzione 50 restituisce il controllo all'istruzione 40 se W ha un valore minore o uguale a zero.

Se viene digitata un'espressione stringa per L o W (per esempio 1.3 Q per W) il sistema visualizza il messaggio d'errore "?Redo from start" e l'utente deve introdurre di nuovo il valore.

L'esecuzione di questo programma continua finchè l'utente non preme **CTRL BREAK**; a questo punto il sistema visualizza il messaggio "Break in *nnnnn*" e ritorna in Stato Comandi. Per riprendere l'esecuzione occorre introdurre:

CONT

INTERRUZIONI DEL PROGRAMMA

Sono possibili tre tipi di interruzioni del programma:

- Interruzioni manuali
- Interruzioni automatiche

• Interruzioni programmabili

SE...	ALLORA...
l'utente preme **CTRL BREAK** (interruzione manuale) o viene eseguita un'istruzione STOP o END (interruzione programmata)	il programma viene interrotto, GW-BASIC entra in Stato Comandi e visualizza Ok. **CTRL BREAK** e STOP non chiudono nessun file dati ma visualizzano il messaggio "Break in *nnnnn*"; END chiude tutti i file dati ma non visualizza tale messaggio. In ogni caso l'utente può riprendere l'esecuzione impostando il comando CONT. L'utente può visualizzare variabili di programma (tramite istruzioni immediate PRINT o PRINT USING) o cambiare i loro valori (tramite istruzioni immediate LET o SWAP). L'utente può anche visualizzare linee di programma tramite un comando EDIT o LIST e modificarle. Se modifica alcune linee, non può continuare l'esecuzione tramite il comando CONT, ma può solo riprendere l'esecuzione del programma digitando RUN.
viene riscontrato un errore sintattico (interruzione automatica)	il programma viene interrotto, GW-BASIC visualizza il messaggio d'errore alla linea che ha causato l'errore, posizionando il cursore sotto la prima cifra del numero di linea. L'utente può modificare la linea e poi eseguire di nuovo il programma impostando RUN. Non può essere usato il comando CONT per continuare l'esecuzione. Se l'utente vuole esaminare i contenuti di alcune variabili prima di fare modifiche deve premere **CTRL BREAK**, per ritornare in Stato Comandi. Dopo aver esaminato i contenuti delle ▶

SE...	ALLORA...
	variabili l'utente può modificare la linea e rieseguire il programma. Per esempio: 10 A=2$6 **RUN** Syntax Error in 10 10 A=2$6
viene riscontrato un errore non sintattico (interruzione automatica)	il programma viene interrotto, GW-BASIC visualizza il messaggio di errore, entra in Stato Comandi e visualizza Ok. L'utente può visualizzare variabili di programma o linee di programma tramite i comandi EDIT o LIST, per poi modificarle. Non può essere usato il comando CONT per continuare l'esecuzione del programma, ma deve essere usato il comando RUN per eseguirlo dall'inizio. Per esempio, l'esecuzione di un programma che contiene: 100 FOR K= causa: Missing operand in 100 Ok
viene riscontrato un errore ed è abilitato il rilevamento di errori	l'esecuzione del programma viene trasferita alla linea specificata dall'istruzione ON ERROR GOTO.

SE...	ALLORA...
(interruzione programmata)	Una routine di rilevamento di errori controlla tutti quei particolari errori che l'utente vuole gestire e specifica il tipo di azione da intraprendere in ogni singolo caso. Questo comprende la correzione dell'errore e la ripresa dell'esecuzione da una istruzione specificata oppure il ritorno allo Stato Comandi.
si verifica un evento e viene abilitato il corrispondente rilevamento di eventi (interruzione programmata)	l'esecuzione del programma viene trasferita alla linea specificata dall'istruzione ON *evento* GOSUB. La routine di rilevamento di eventi specifica il tipo di azione da intraprendere in questo caso. Gli eventi che possono essere rilevati sono la ricezione di caratteri da una porta di comunicazione, la pressione di un tasto, il passare del tempo o lo svuotamento del buffer per la musica di sottofondo. Si faccia riferimento al Capitolo 6 (RILEVAMENTO DI EVENTI) per ulteriori dettagli.

Esempio

```
 10 ON ERROR GOTO 100
 20 INPUT "WHAT IS YOUR BET"; B
 30 IF B>5000 THEN ERROR 200
  .
  .
  .
100 IF ERR=200 THEN PRINT "HOUSE LIMIT IS $5000"
110 IF ERL=30 THEN RESUME 20
120 ON ERROR GOTO 0
```

Se l'utente imposta un valore di B > 5000 viene visualizzato il messaggio:

HOUSE LIMIT IS $5000

e l'esecuzione riprende dalla linea 20.

L'istruzione 10 (ON ERROR GOTO 100) abilita il rilevamento di errori e trasferisce il controllo alla linea 100 se viene riscontrato un errore. L'istruzione ERROR 200 alla linea 30 genera l'errore definito dall'utente, cioè 200. Quando questa istruzione viene eseguita il controllo viene trasferito alla linea 100.

Nell'istruzione 120, la clausola ON ERROR GOTO 0 disabilita il rilevamento di errori, perciò se viene riscontrato un altro errore diverso dall'errore 200 viene visualizzato il messaggio di errore standard.

Si noti il significato delle funzioni speciali di sistema ERR e ERL; la prima contiene il codice d'errore, la seconda il numero di linea sulla quale è stato riscontrato l'errore.

3. COSTANTI, VARIABILI ED ESPRESSIONI

SOMMARIO

Questo capitolo evidenzia gli elementi principali che possono essere introdotti in una linea GW-BASIC: costanti, variabili, espressioni ed operatori.

INDICE

COSTANTI

Le costanti sono i valori utilizzati da GW-BASIC durante l'esecuzione dei programmi. Ci sono due tipi di costanti: costanti stringa e costanti numeriche.

Una costante stringa è una sequenza di caratteri alfanumerici racchiusi tra doppi apici la cui lunghezza massima è 255 caratteri.

Per esempio:

"READY"
"$80"
"accelleration rate"

Le costanti numeriche sono numeri positivi o negativi. Le costanti numeriche GW-BASIC non possono contenere virgole. Ci sono cinque tipi di costanti numeriche:

1. Costanti intere. Sono costituite da tutti i numeri compresi tra -32768 e 32767; non contengono punti decimali.

2. Costanti a virgola fissa. Sono costituite da numeri reali positivi o negativi cioè numeri che contengono il punto decimale.

3. Costanti a virgola mobile. Sono costituite da numeri positivi o negativi rappresentati in forma esponenziale. Una costante a virgola mobile è costituita da un intero con o senza segno (la mantissa) seguito dalla lettera E e da un intero con o senza segno (l'esponente). Le costanti a virgola mobile sono comprese tra 10E-38 e 10E+38.

   Esempi:

   235.988E-7 è equivalente a .0000235988
   2359E6 è equivalente a 2359000000.

   Le costanti a virgola mobile in doppia precisione vengono indicate dalla lettera D invece di E.

4. Costanti esadecimali. Sono costituite da numeri esadecimali indicati dal prefisso &H.

   Esempi:

   &H76
   &H32F
   &HFFAA

5. Costanti in base 8. Sono costituite da numeri in base 8 indicati dal prefisso &O o &.

   Esempi:

   &O347
   &1234

RAPPRESENTAZIONE DI COSTANTI NUMERICHE IN SINGOLA E DOPPIA PRECISIONE

Le costanti numeriche possono essere numeri in singola o in doppia precisione. Le costanti numeriche in singola precisione vengono memorizzate e stampate con 7 cifre. Le costanti numeriche in doppia precisione vengono memorizzate con 16 cifre e stampate con un massimo di 16.

Una costante in singola precisione è una qualsiasi costante numerica che possiede una delle seguenti caratteristiche:

1. Sette cifre, o meno

2. Forma esponenziale tramite E

3. Un punto esclamativo (!) finale

Per esempio:

46.8
-1.09E-06
3489.0
22.5!

Una costante in doppia precisione è una qualsiasi costante numerica che possiede una delle seguenti caratteristiche:

1. Otto cifre, o più
2. Forma esponenziale tramite D
3. Un segno di numero (#) finale

Per esempio:

345692811
-1.094320-06
3489.0#
7654321.1234

VARIABILI

Le variabili sono nomi che rappresentano valori usati in un programma GW-BASIC. Il valore di una variabile può essere determinato esplicitamente dal programmatore o può essere assegnato come risultato di calcoli fatti nel programma. Prima di assegnare un valore a una variabile, si suppone che la variabile contenga valore zero.

NOMI DI VARIABILI E CARATTERI DI DICHIARAZIONE

I nomi di variabili GW-BASIC possono essere di qualsiasi lunghezza. Essi sono significativi fino a 40 caratteri. I nomi di variabili possono contenere lettere, numeri e il punto decimale; comunque il primo carattere deve essere una lettera. Sono anche ammessi caratteri speciali di dichiarazione di tipo.

Un nome di variabile non può essere una parola riservata, ma può contenere parole riservate. Le parole riservate comprendono tutti i comandi, le istruzioni, i nomi delle funzioni e i nomi degli operatori GW-BASIC. Una variabile che inizia con FN indica una chiamata a una funzione definita dall'utente.

Le variabili possono rappresentare un valore numerico o una stringa. I nomi delle variabili stringa vengono scritti con un segno

dollaro ($) come ultimo carattere. Per esempio: A$ = "SALES REPORT". Il segno dollaro è un carattere di dichiarazione del tipo della variabile e cioè dichiara che la variabile rappresenta una stringa.

I nomi delle variabili numeriche possono dichiarare valori interi in singola o in doppia precisione. I caratteri di dichiarazione di tipo per questi nomi di variabile sono i seguenti:

% Variabile intera

! Variabile in semplice precisione

Variabile in doppia precisione

Il tipo di default per una variabile numerica è semplice precisione.

Esempi di Nomi di Variabili GW-BASIC

PI#	dichiara un valore in doppia precisione.
MINIMUM!	dichiara un valore in singola precisione.
LIMIT%	dichiara un valore intero.
N$	dichiara un valore stringa.
ABC	rappresenta un valore in singola precisione.

C'è un secondo metodo per dichiarare tipi di variabile. Le istruzioni GW-BASIC DEFINT, DEFSTR, DEFSNG e DEFDBL possono essere incluse in un programma per dichiarare i tipi di certe variabili. Queste istruzioni vengono descritte dettagliatamente nel Capitolo 8.

MATRICI

Una matrice è un insieme di variabili con lo stesso nome. Ogni variabile (o elemento della matrice) può essere riferito specificando uno o piú interi o espressioni intere (indici), posti tra parentesi dopo il nome della matrice. Per esempio V(10) indica un

elemento di una matrice a una dimensione o vettore, T(1,4) indica un elemento di una matrice a due dimensioni e così via. Una matrice può avere al massimo 255 dimensioni, e ogni dimensione può avere al massimo 32767 elementi. Entrambi questi valori vengono anche limitati dalle dimensioni effettive della memoria del sistema.

In seguito, ogni volta che un parametro indica una variabile, nella sintassi di un'istruzione o di un comando, può riferirsi sia a una variabile semplice che a un elemento di matrice. Se sarà necessario distinguerli verrà specificato nel contesto.

OCCUPAZIONE DI MEMORIA

Di seguito viene elencato il numero di byte occupati da stringhe, variabili e matrici.

Tipo di variabile	Byte
Intero	2
In singola precisione	4
In doppia precisione	8

Tipo di matrice	Byte
Intero	2 per elemento
In singola precisione	4 per elemento
In doppia precisione	8 per elemento

Stringhe

Richiedono 3 byte aggiuntivi oltre quelli necessari per la memorizzazione della stringa.

CONVERSIONE DI TIPO

Se è necessario, GW-BASIC converte una costante numerica da un tipo all'altro, si considerino le regole e gli esempi seguenti:

1. Se una costante numerica di un tipo viene assegnata a una variabile numerica di tipo diverso, il numero viene memorizzato secondo il tipo dichiarato per la variabile. Se a una variabile stringa viene assegnato un valore numerico o viceversa, viene visualizzato il messaggio di errore "Type mismatch" (Incompatibilità di tipo).

   Esempio:

   ```
   10 A% = 23.42
   20 PRINT A%
   RUN
    23
   ```

2. Durante il calcolo di un'espressione tutti gli operandi in un'operazione aritmetica o di confronto vengono arrotondati allo stesso livello di precisione, cioè a quello dell'operando più preciso. Anche il risultato di un'operazione aritmetica viene arrotondato a questo livello di precisione.

   Esempio:

   ```
   10 D# = 6#/7
   20 PRINT D#
   RUN
    .8571428571428571
   ```

   I calcoli aritmetici vengono eseguiti in doppia precisione e il risultato viene assegnato a D# come valore in doppia precisione.

   ```
   10 D = 6#/7
   20 PRINT D
   RUN
    .8571429
   ```

   I calcoli aritmetici vengono eseguiti in doppia precisione e il risultato viene assegnato a D (variabile in singola precisione), arrotondato e stampato come valore in singola precisione.

3. Gli operatori logici convertono i loro operandi in interi e restituiscono un risultato intero. Gli operandi devono essere compresi tra -32768 e 32767, altrimenti viene visualizzato il messaggio d'errore "Overflow". Una completa descrizione degli operatori logici viene fornita più avanti in questo capitolo.

4. Quando un valore a virgola mobile viene convertito in intero, la parte frazionaria viene arrotondata.

   Esempio:

   ```
   10 C% = 55.88
   20 PRINT C%
   RUN
    56
   ```

5. Se ad una variabile in doppia precisione viene assegnato un valore in singola precisione, solo le prime sette cifre (arrotondate) del numero convertito sono valide. Questo avviene perchè solo sette cifre di precisione sono state sostituite con il valore in singola precisione. Il valore assoluto della differenza tra il numero in doppia precisione stampato e il valore in singola precisione originale è inferiore a 6.3E-8 volte il valore in singola precisione originale.

   Esempio:

   ```
   10 A = 2.04
   20 B# = A
   30 PRINT A;B#
   RUN
    2.04 2.039999961853027
   ```

ESPRESSIONI E OPERATORI

Un'espressione può essere costituita da una costante stringa, una costante numerica, una variabile o una combinazione di costanti e variabili con operatori. Un'espressione produce sempre un unico valore. Gli operatori eseguono operazioni matematiche o logiche sui valori.

Gli operatori GW-BASIC possono essere suddivisi in quattro tipi:

1. Aritmetici

2. Di confronto

3. Logici

4. Funzionali

Ogni tipo viene descritto nelle sottosezioni che seguono.

OPERATORI ARITMETICI

Gli operatori aritmetici sono riportati nella Tabella 3-1 in ordine di precedenza

OPERATORE	OPERAZIONE	ESEMPIO
^	Elevamento a potenza	X ^ Y
-	Cambiamento di segno	-X
*,/	Moltiplicazione, Divisione a punto-mobile	X*Y X/Y
\	Divisione tra interi	X \ Y
MOD	Divisione Modulo	X MOD Y
+,-	Addizione, Sottrazione	X+Y, X-Y

Tab. 3-1 Operatori Aritmetici

Per cambiare l'ordine in cui le operazioni vengono eseguite devono essere usate le parentesi. Le operazioni tra parentesi vengono eseguite per prime. All'interno delle parentesi viene mantenuto l'ordine normale delle operazioni.

Quelle che seguono sono alcune espressioni algebriche, con la loro rappresentazione in GW-BASIC.

Espressione Algebrica	Espressione GW-BASIC
$X+2Y$	X+Y*2
$X-\frac{Y}{Z}$	X-Y/Z
$\frac{XY}{Z}$	X*Y/Z
$\frac{X+Y}{Z}$	(X+Y)/Z
$(X^2)^Y$	(X ^ 2) ^ Y
X^{Y^Z}	X ^ (Y ^ Z)
X(-Y)	X*(-Y)

Nota

Due operatori consecutivi devono essere separati da parentesi, come nell'esempio X*(-Y).

DIVISIONE TRA INTERI E DIVISIONE MODULO

Sono disponibili in GW-BASIC altri due operatori: divisione tra interi e divisione modulo.

La divisione tra interi viene indicata dalla barra inversa (\). Gli operandi vengono arrotondati al numero intero più vicino; prima di effettuare la divisione il quoziente viene troncato al valore intero più vicino. Gli operandi devono essere compresi tra -32768 e 32767.

Esempio

10\4 = 2
25.68\6.99 = 3

La divisione tra interi segue, in ordine di precedenza, la moltiplicazione e la divisione tra numeri a virgola mobile.

Il modulo viene indicato dall'operatore MOD. Il modulo fornisce il numero intero che è il resto di una divisione tra numeri interi.

Esempio

10.4 MOD 4 = 2 (10/4 = 2 con resto 2)
25.68 MOD 6.99 = 5 (26/7 = 3 con resto 5)

Il modulo segue, in ordine di precedenza, la divisione tra interi.

OVERFLOW

Se, durante il calcolo di un'espressione, viene incontrata una divisione per zero viene visualizzato il messaggio di errore "Division by zero"; come risultato della divisione viene fornito l'infinito di macchina (il numero più alto che può essere rappresentato in virgola mobile) con il segno del numeratore, dopodichè l'esecuzione continua. Se il risultato di un elevamento a potenza è zero elevato a una potenza negativa, viene nuovamente visualizzato il messaggio d'errore "Division by zero", come risultato dell'elevamento a potenza viene fornito l'infinito di macchina positivo e l'esecuzione continua.

Se si verifica un overflow viene visualizzato il messaggio d'errore "Overflow", viene fornito come risultato l'infinito di macchina con l'opportuno segno algebrico e l'esecuzione continua.

OPERATORI DI CONFRONTO

Gli operatori di confronto vengono usati per confrontare due valori. Il risultato del confronto è "vero" (-1) o "falso" (0). Questo risultato può essere usato per prendere una decisione riguardo al flusso del programma.

OPERATORE	RELAZIONE	ESEMPIO
=	Uguaglianza	X=Y
<>(o ><)	Disuguaglianza	X<>Y
<	Minore di	X<Y
>	Maggiore di	X>Y
<=	Minore o uguale a	X<=Y
>=	Maggiore o uguale a	X>=Y

Tab. 3-2 Operatori di Confronto

Il segno di uguaglianza (=) viene usato anche per assegnare un valore a una variabile.

Quando in una stessa espressione sono presenti operatori aritmetici e di confronto l'operazione aritmetica viene sempre eseguita per prima. Per esempio l'espressione:

X+Y<(T-1)/Z

è vera se la somma di X e Y è inferiore al valore di (T-1)/Z.

Altri esempi:

```
IF SIN(X)<0 GOTO 1000
IF I MOD J<>0 THEN K=K+1
```

OPERATORI LOGICI

Gli operatori logici eseguono test su relazioni multiple, elaborazioni di bit o operatori Booleani. L'operatore logico restituisce un valore che può essere "vero" (non zero) o "falso" (zero). In una espressione le operazioni logiche vengono eseguite dopo le

operazioni aritmetiche e di confronto. Il risultato di un'operazione logica è determinato come illustrato nella Figura 3-3.

Gli operatori sono elencati in ordine di precedenza.

NOT

X	NOT X
1	0
0	1

AND

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

OR

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

XOR

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

Fig. 3-3 Tavola di Verità degli Operatori Logici GW-BASIC (cont.)

EQV

X	Y	X EQV Y
1	1	1
1	0	0
0	1	0
0	0	1

IMP

X	Y	X IMP Y
1	1	1
1	0	0
0	1	1
0	0	1

Fig. 3-3 Tavola di Verità degli Operatori Logici GW-BASIC

Come gli operatori di confronto possono essere usati per prendere decisioni riguardanti il flusso del programma, così gli operatori logici possono connettere due o più relazioni e restituire un valore "vero" o "falso" da usare in una decisione successiva.

Esempio

```
IF D<200 AND F<4 THEN 80
IF I>10 OR K<0 THEN 50
IF NOT P THEN 100
```

Gli operatori logici convertono i loro operandi in stringhe di 16 bit con segno, usando la rappresentazione intera complemento a due nell'intervallo che va da -32768 a 32767 (quando gli operandi si trovano fuori da questo intervallo si verifica un errore). Se entrambi gli operandi vengono indicati come 0 o -1, gli operatori logici restituiscono 0 o -1. L'operazione viene svolta su questi numeri interi bit per bit, cioè ogni bit del risultato è determinato dai bit corrispondenti dei due operandi.

Perciò è possibile usare operatori logici per controllare i byte di una particolare configurazione di bit. Per esempio, l'operatore AND può essere usato per "mascherare" tutti i bit, eccetto uno, di un byte di stato in una porta di I/O. L'operatore OR può essere usato per fare il "merge" di due byte in modo da creare un particolare valore binario. Gli esempi che seguono illustrano il modo in cui agiscono gli operatori logici.

```
63 AND 16=16       il risultato è 16, poichè
                   63 = in binario        111111
                   16 = in binario        010000
                                          ------
                                          010000

15 AND 14=14       il risultato è 14, poichè
                   15 = in binario        1111
                   14 = in binario        1110
                                          ----
                                          1110

-1 AND 8=8         il risultato è 8, poichè
                   -1 = in binario        1111111111111111
                   8 = in binario         0000000000001000
                                          ----------------
                                          0000000000001000

4 OR 2=6           il risultato è 6, poichè
                   4 = in binario         100
                   2 = in binario          10
                                          ---
                                          110

10 OR 10=10        il risultato è 10, poichè
                   10 = in binario        1010
                   10 = in binario        1010
                                          ----
                                          1010

-1 OR -2=-1        il risultato è -1, poichè
                   -1 = in binario        1111111111111111
                   -2 = in binario        1111111111111110
                                          ----------------
                                          1111111111111111
```

NOT X = -X + 1	Il complemento a due di un intero si ottiene invertendo tutti i bit del numero (0→1, 1→0) e sommando 1 al risultato così ottenuto.

Esempio

1111100110011000 valore originale (negativo)

Invertendo tutti i bit:

0000011001100111 valore invertito

Aggiungendo 1:

0000011001101000 valore assoluto (invertito + 1)

Il valore decimale della configurazione data è quindi:

1640

OPERATORI FUNZIONALI

Quando una funzione viene usata in un'espressione richiama un'operazione predeterminata che deve essere svolta su un operando. GW-BASIC ha delle funzioni "intrinseche" che risiedono nel sistema, come per esempio SQR (radice quadrata) o SIN (seno). Tutte le funzioni intrinseche GW-BASIC vengono descritte nel Capitolo 8.

GW-BASIC permette anche funzioni "definite dall'utente" che vengono scritte dal programmatore (istruzioni DEF FN).

OPERATORI STRINGA

Le stringhe possono essere concatenate tra loro mediante il segno di addizione (+).

Esempio

```
10 A$ = "FILE" : B$ = "NAME"
20 PRINT A$ + B$
30 PRINT "NEW " + A$ + B$
RUN
FILENAME
NEW FILENAME
```

Le stringhe possono essere confrontate tramite gli stessi operatori usati per i numeri:

= <> < > <= (o =<) >= (o =>)

I confronti di stringhe vengono fatti confrontando i codici ASCII dei caratteri corrispondenti. Se tutti i confronti risultano uguali anche le stringhe sono uguali. Se in uno dei confronti i caratteri sono diversi, la stringa maggiore è quella il cui carattere ha valore maggiore secondo la codifica ASCII. Se durante il confronto viene raggiunta la fine di una stringa, si dice che la stringa più corta ha un valore minore dell'altra. Gli spazi di testa e di coda sono significativi.

Esempio

```
"AA"<"AB"
"FILENAME"="FILENAME"
"X&">"X="
"CL ">"CL"
"kg">"KG"
"SMYTH"<"SMYTHE"
B$>"9/12/78"          dove B$="8/12/78"
```

I confronti tra stringhe possono essere usati per mettere in ordine alfabetico le stringhe. Tutte le costanti stringa usate nelle espressioni di confronto devono essere tra doppi apici.

4. GESTIONE DI FILE SU DISCO

SOMMARIO

Questo capitolo descrive come creare e gestire i file su disco; fornisce inoltre una descrizione completa delle loro prestazioni e funzioni.

INDICE

INTRODUZIONE

Questo capitolo descrive le procedure di input e output su disco che l'utente GW-BASIC deve eseguire. Le procedure e gli esempi di programma contenuti in questo capitolo sono diretti ad un utente che non conosce ancora il linguaggio GW-BASIC o che ha commesso errori relativi al disco, in modo da verificare che tutte le istruzioni su disco vengano usate in modo corretto.

Quando viene richiesto un nome di file in un comando o in un'istruzione è necessario usare un nome conforme alle convenzioni usate per l'identificazione dei file. Il sistema operativo MS-DOS provvede ad aggiungere l'estensione di default .BAS al nome del file specificato in un comando SAVE, RUN, MERGE o LOAD.

INPUT/OUTPUT INDIPENDENTE DAL DEVICE

GW-BASIC fornisce una prestazione di input/output indipendente dal device; usare tale prestazione significa avere la stessa sintassi di accesso per qualsiasi device.

Le seguenti istruzioni, comandi e funzioni vengono usate per la prestazione di I/O indipendente dal device:

BLOAD
BSAVE
CHAIN
CLOSE
EOF
GET
INPUT#
INPUT$
LINE INPUT#
LIST
LOAD
LOC
LOF
MERGE
OPEN
POS
PRINT#
PRINT#USING
PUT
RUN
SAVE
WIDTH
WRITE#

COME MS-DOS CATALOGA I FILE

I nomi dei file vengono catalogati in directory su disco. Queste directory contengono anche informazioni sulla dimensione dei file, sulla loro allocazione su disco e le date di creazione e di aggiornamento. La directory su cui l'utente sta lavorando viene chiamata directory corrente.

Un'area supplementare di sistema, chiamata Tabella di Allocazione dei File, cataloga la locazione dei file sul disco ed alloca anche lo spazio libero in modo da permettere la creazione di nuovi file.

Queste due aree di sistema, directory e Tabella di Allocazione dei File, permettono a MS-DOS di riconoscere ed organizzare i file su disco. La Tabella di Allocazione dei File viene creata su un nuovo disco quando l'utente lo formatta con il comando MS-DOS FORMAT, si crea così una directory vuota, denominata directory "root" (radice).

Per usare le informazioni bisogna aprire il file tramite l'istruzione OPEN per comunicare a GW-BASIC dove si trova l'informazione. L'utente può quindi usare il file per le operazioni di input e/o output.

NUMERI DI FILE

Un numero di file è costituito da un intero compreso tra 1 e 255, associato ad un file al momento della sua apertura, e viene usato da qualsiasi istruzione successiva di I/O per fare riferimento al file. Durante l'esecuzione di un programma GW-BASIC il numero massimo di file che possono essere aperti simultaneamente viene stabilito dallo switch /F: nel comando GWBASIC.

COME ASSEGNARE UN NOME AI FILE

Le specificazioni di file (o filescpec) seguono le regole di MS-DOS per l'assegnazione di nomi. *filespec* è un'espressione stringa con il seguente formato:

[*device*:] *filename*

Tutte le specificazioni di file possono iniziare con una dichiarazione di device come per esempio A: oppure B: oppure COM1: oppure LPT1:. Se non viene specificato alcun device, viene assunto il device attuale.

Un nome di file può comprendere:

- da uno a otto caratteri; per esempio NEWFILE
- da uno a otto caratteri, seguiti da un punto (.) e da uno a tre caratteri di estensione; per esempio: NEWFILE.EXE.

Un nome di file può essere composto da qualsiasi dei seguenti caratteri:

A-Z   0-9 $ &   #   ~

%   '   (   )   -   _

@   ^   {   }   !

I caratteri alfabetici all'interno del nome di file possono essere digitati in lettere maiuscole o minuscole, MS-DOS trasforma le lettere minuscole in lettere maiuscole. Se non è stata fornita alcuna estensione quella di default è .BAS, ma i comandi NAME e KILL non seguono questa regola e non forniscono alcuna estensione.

Nota

Per specificare un file quando viene usata l'istruzione "OPEN COM..." (per aprire un device di comunicazione) viene usato un formato diverso.

Il nome di file è sostituito da una lista di opzioni che specificano elementi come la velocità di linea, la parità, il numero di bit di dati per ogni byte, il numero di bit di stop, ecc. Si faccia riferimento all'istruzione OPEN COM del Capitolo 8 per ulteriori dettagli.

Quando si usa una costante stringa per specificare un file, è necessario racchiuderla tra doppi apici. L'unica eccezione a questa regola è il comando GWBASIC, dove *filespec* è una costante stringa non racchiusa tra doppi apici.

Per esempio:

LOAD "B:ARSENAL.RED"

Per esempio:

GWBASIC PAYROLL

COME ASSEGNARE UN NOME AI DEVICE

Il nome di un device è una stringa composta da non più di quattro caratteri seguita dai due punti (:), può essere uno dei seguenti:

A: primo drive di diskette	(Qualsiasi modo di accesso)
B: secondo drive di diskette	(Qualsiasi modo di accesso)
C: primo drive di hard disk	(Qualsiasi modo di accesso)
D: secondo drive di hard disk	(Qualsiasi modo di accesso)
KYBD: tastiera	(Solo Input)
SCRN: video	(Solo Output)
LPT1: prima stampante	(Output o Diretto)
LPT2: seconda stampante	(Output o Diretto)
LPT3: terza stampante	(Output o Diretto)

COM1: Porta RS232 N.1 (Input o Output)
COM2: Porta RS232 N. 2 (Input o Output)

I PATHNAME

Con GW-BASIC l'utente può organizzare un disco in modo tale da utilizzare solo i file richiesti dall'attività in corso.

In passato, veniva utilizzata una sola directory per contenere tutti i file residenti su uno stesso disco. MS-DOS estende questo concetto permettendo ad una directory di contenere sia file che directory e introduce il concetto di directory corrente.

Per specificare un file, l'utente può specificare sia un cammino dalla directory radice (root) al file sia un cammino dalla directory corrente al file. Un pathname è una sequenza di nomi di directory, separati da '\', che termina con un nome di file. Un pathname che parte dalla radice inizia con '\'.

Per ogni directory sono fornite due speciali abbreviazioni caratterizzate dai segni '.' e '..'. Questi servono a specificare la directory corrente (.) e la directory madre (..).

Per esempio, in una certa attività commerciale, sia le vendite che la contabilità utilizzano uno stesso computer con un disco di grande capacità, inoltre i singoli impiegati lo usano per la preparazione di resoconti e per mantenere aggiornate le informazioni sulla contabilità.

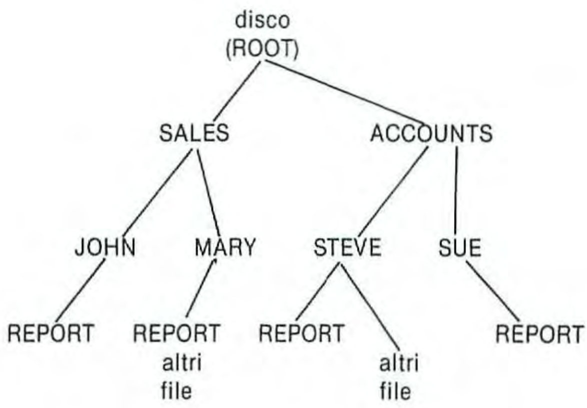


Fig. 4-1 Organizzazione dei File su Disco

Usando per la directory una struttura gerarchica, come quella vista sopra, e presupponendo che la directory corrente sia la directory JOHN, per fare riferimento a REPORT sotto JOHN, le due notazioni che seguono sono equivalenti:

REPORT

\SALES\JOHN\REPORT

Per fare riferimento a REPORT sotto MARY, considerando JOHN directory corrente, le due notazioni che seguono sono equivalenti:

..\MARY\REPORT

\SALES\MARY\REPORT

Per fare riferimento a REPORT sotto SUE, considerando JOHN directory corrente, le due notazioni che seguono sono equivalenti:

..\..\ACCOUNTS\SUE\REPORT

\ACCOUNTS\SUE\REPORT

Non ci sono limiti alle ramificazioni di un albero. La directory radice ha un numero massimo di file e/o sottodirectory su un dischette. Non c'è alcun limite per il numero di file e/o di sottodirectory nella directory radice di un hard disk, tranne che per la dimensione della partizione MS-DOS (vedere "MS-DOS Guida Utente").

È possibile anche accedere ad altre sottodirectory, tramite la directory radice, e queste, a loro volta, possono dare origine ad ulteriori file e sottodirectory. L'unico limite è lo spazio disponibile sul disco.

Ogni directory può anche contenere file e sottodirectory con lo stesso nome dei file e delle sottodirectory che appartengono ad un'altra directory.

Si possono utilizzare pathname con i seguenti comandi:

BLOAD	GWBASIC(*)	NAME
BSAVE	KILL	OPEN
CHAIN	LOAD	RMDIR
CHDIR	MERGE	RUN
FILES	MKDIR	SAVE

(*) Viene usato per attivare l'ambiente GW-BASIC. Questo è un comando di MS-DOS (non un comando di GW-BASIC).

Un pathname può essere considerato come un'estensione di *filespec* ed è un'espressione stringa della forma:

[*device*:][\][*directory*][*directory*]...[\] *filename*

o

[*device*:][\][*directory*][*directory*]...[*directory*][\]

Tutti i caratteri che possono essere usati per specificare un nome di file possono essere utilizzati anche per una directory.

Esempio (supposto JOHN come directory corrente); le espressioni sequenti sono equivalenti:

```
B:\SALES\MARY\REPORT
B:..\MARY\REPORT
```

Nota

Un pathname non può contenere più di 63 caratteri. I pathname più lunghi di 63 caratteri vengono segnalati con l'errore "Bad filename".

Se viene specificato un pathname in cui un solo filespec è valido oppure se viene indicato un device in una posizione diversa da quella iniziale del pathname viene emesso l'errore "Bad filename".

Se per il pathname viene usata una costante stringa, questa deve essere racchiusa tra doppi apici; solo il comando GWBASIC specifica i pathname come stringhe non racchiuse tra doppi apici.

LA DIRECTORY CORRENTE

Se in un'istruzione o in un comando di GW-BASIC viene introdotto un nome di file, senza che sia stata specificata la directory, viene ricercata la directory corrente. All'atto della formattazione, viene creata su disco una singola directory, chiamata directory radice, che inizialmente è la directory corrente. È possibile creare altre directory tramite il comando MKDIR oppure rimuovere le directory con il comando RMDIR. Il comando CHDIR pemette all'utente di cambiare la directory corrente.

Se un pathname è preceduto da una barra rovesciata (\), GW-BASIC inizia la sua ricerca partendo dalla directory radice, altrimenti dalla directory corrente. Il pathname specificato dall'utente può essere una sequenza di nomi di directory che incominciano con la radice o con la directory corrente. Se il file appartiene alla directory corrente, basta specificare il file.

COMANDI PER FILE DI PROGRAMMI

L'elenco che segue riassume le istruzioni e i comandi usati nella gestione dei file contenenti i programmi.

In GW-BASIC l'asterisco (*) e il punto interrogativo (?) possono essere usati come caratteri "Jolly" con i comandi FILES e KILL.

SINTASSI	SIGNIFICATO
SAVE *filespec* [,{A I P}] o SAVE *pathname* [,{A I P}]	Registra su disco il programma attualmente residente in memoria. Con l'opzione A il programma viene registrato come una serie di caratteri ASCII (altrimenti GW-BASIC usa un formato binario impaccato); con l'opzione P il programma viene registrato in forma protetta (vedere "FILE PROTETTI" in questo capitolo).
LOAD *filespec* [,R] o LOAD *pathname* [,R]	Trasferisce il programma da disco in memoria. Con l'opzione R il programma viene eseguito immediatamente. Il cocomando LOAD cancella sempre il contenuto corrente della memoria e chiude tutti i file prima di effettuare il trasferimento. Se viene inclusa l'opzione R, i file dati aperti vengono mantenuti aperti. Così, i programmi possono essere concatenati o introdotti in sezioni e possono

SINTASSI	SIGNIFICATO
	accedere agli stessi file dati. Gli elementi di sintassi LOAD *filespec*, R e RUN *filespec*, R sono equivalenti.
RUN *filespec* [,R] o RUN *pathname* [,R]	RUN *filespec* trasferisce il programma da disco in memoria e lo esegue. RUN cancella il contenuto corrente della memoria e chiude tutti i file prima di trasferire il programma. Se viene inclusa l'opzione R tutti i file dati aperti vengon mantenuti aperti.
MERGE *filespec* o MERGE *pathname*	Inserisce le linee del programma su disco tra le linee del programma residente in memoria. Se due linee hanno lo stesso numero, viene conservata solo la linea del programma su disco. Dopo il comando MERGE, il programma risultante risiede in memoria e GW-BASIC ritorna in stato comandi.
KILL *filespec* o KILL *pathname*	Cancella il file su disco. Il file può essere un file programma o un file dati ad accesso sequenziale o diretto.
NAME {*filespec* I *pathname*} AS *filename*	Cambia il nome di un file su disco. NAME può essere usato con qualsiasi tipo di file su disco.

FILE PROTETTI

Se l'utente vuole registrare un programma in un formato binario codificato, deve usare l'opzione P con il comando SAVE. Per esempio:

```
SAVE "MYPROG",P
```

Nota

Poiché un programma registrato in questo modo non può essere listato o editato, l'utente può decidere di registrare una copia non protetta del programma a questo scopo.

FILE DATI SU DISCO — ACCESSO SEQUENZIALE E DIRETTO

Un programma GW-BASIC può creare ed accedere a due tipi di file dati su disco:

- File sequenziali
- File ad accesso diretto

FILE SEQUENZIALI

I file sequenziali sono più semplici da creare ma riducono la flessibilità e la velocità di accesso ai dati.

I dati scritti su un file sequenziale sono in formato ASCII e vengono caricati e memorizzati, uno dopo l'altro (in sequenza), nell'ordine in cui sono stati inviati.

Le istruzioni e le funzioni usate con i file sequenziali sono le seguenti:

CLOSE
EOF
INPUT$
INPUT#
LINE INPUT#
LOC
LOCK
LOF
OPEN
PRINT#
PRINT#USING
UNLOCK
WRITE#

Si faccia riferimento al Capitolo 8 per ulteriori informazioni su queste istruzioni e funzioni.

COME CREARE UN FILE SEQUENZIALE

Per creare un file sequenziale ed accedere ai dati in esso contenuti sono richieste le seguenti fasi di programma:

1. Aprire il file con modalità "O"

    ```
    10 OPEN "O", #1, "DATA"
    ```

2. Scrivere i dati sul file tramite l'istruzione WRITE# (si può anche usare l'istruzione PRINT#)

    ```
    20 WRITE#1,A$;B$;C$
    ```

3. Per accedere ai dati sul file, l'utente deve chiudere (CLOSE) il file e riaprirlo (OPEN) con modalità "I"

    ```
    50 CLOSE#1
    ```

    ```
    60 OPEN "I",#1,"DATA"
    ```

4. Usare l'istruzione INPUT# per leggere i dati dal file sequenziale

```
70 INPUT #1,X$,Y$,Z$
```

Un programma che crea un file sequenziale può anche registrare dati formattati su disco tramite l'istruzione PRINT#USING. Per esempio, l'istruzione

```
80 PRINT#1,USING"####.##,";A,B,C,D
```

può essere usata per scrivere su disco dati numerici senza delimitatori espliciti. La virgola (,) alla fine della stringa di formato serve per separare ogni voce presente sul file su disco.

La funzione LOC, quando viene usata con un file sequenziale, riporta il numero di settori che sono stati scritti sul file o letti dal file dal momento in cui esso è stato aperto. Per esempio:

```
100 IF LOC(1)>50 THEN STOP
```

termina l'esecuzione del programma se più di 50 settori sono stati scritti o letti dal file #1 dal momento in cui esso è stato aperto.

Il programma 1 è un breve programma che crea un file sequenziale, chiamato "DATA", in seguito alle informazioni che l'utente ha introdotto da terminale.

```
10 OPEN "O",#1,"DATA"
20 INPUT "NAME";N$
25 IF N$="DONE" THEN END
30 INPUT "DEPARTMENT";D$
40 INPUT "DATE HIRED";H$
50 PRINT#1,N$;",";D$;",";H$
60 PRINT:GOTO 20
RUN
NAME? MICKEY MOUSE
DEPARTMENT? AUDIO/VISUAL AIDS
DATE HIRED? 01/12/72
```

Fig. 4-2 Programma 1 - Creazione di un File Dati Sequenziale (cont.)

```
NAME? SHERLOCK HOLMES
DEPARTMENT? RESEARCH
DATE HIRED? 12/03/65

NAME? EBENEEZER SCROOGE
DEPARTMENT? ACCOUNTING
DATE HIRED? 04/27/78

NAME? SUPER MAN
DEPARTMENT? SECURITY
DATE HIRED? 08/16/78

NAME? ecc.
```

Fig. 4-2 Programma 1 - Creazione di un File Dati Sequenziale

COME ACCEDERE AD UN FILE SEQUENZIALE

Il programma 2 permette l'accesso al file "DATA" che è stato creato nel programma 1 e visualizza il nome di tutti coloro che sono stati assunti nel 1978.

```
10 OPEN "I",#1,"DATA"
20 INPUT#1,N$,D$,H$
30 IF RIGHT$(H$,2)="78" THEN PRINT N$
40 GOTO 20
RUN
EBENEEZER SCROOGE
SUPER MAN
Input past end in 20
Ok
```

Fig. 4-3 Programma 2 - Accesso a un File Sequenziale

Il programma 2 legge, in sequenza, ogni elemento del file. Quando tutti i dati sono stati letti, la linea 20 provoca l'errore "Input past end". Questo errore può essere evitato tramite l'inserimento di una linea supplementare (la linea 15 illustrata sotto) che usa la funzione EOF per controllare la condizione di end-of-file.

```
15 IF EOF(1) THEN END
```

Quindi, cambiare la linea 40 in GOTO 15.

COME AGGIUNGERE DATI AD UN FILE SEQUENZIALE

Non appena un file sequenziale viene aperto su disco con modalità "O", il suo contenuto corrente viene distrutto. Per aggiungere un numero maggiore di dati al file è necessario usare l'istruzione OPEN con modalità APPEND.

FILE AD ACCESSO DIRETTO

La creazione e l'accesso a file ad accesso diretto richiedono più fasi di quelle necessarie per i file sequenziali, ma rispetto a questi offrono dei vantaggi. I file ad accesso diretto richiedono meno spazio sul disco, poiché GW-BASIC li catagola in formato binario impaccato (un file sequenziale è memorizzato come una serie di caratteri ASCII).

Il maggiore vantaggio dei file ad accesso diretto consiste nel fatto che si può accedere ai dati in modo diretto, cioé in qualsiasi punto del disco. Non è necessario, con i file ad accesso diretto, leggere tutte le informazioni presenti sul disco come con i file sequenziali. Questo è possibile, poiché l'informazione è memorizzata ed accessibile in unità distinte chiamate record e ogni record ha un numero di identificazione.

Le istruzioni e le funzioni che vengono usate con i file ad accesso diretto sono:

CLOSE
CVD
CVI
CVS.
FIELD
GET
LOC
LOCK
LOF
LSET
MKD$
MKI$
MKS$
OPEN
PUT
RSET
UNLOCK

COME CREARE UN FILE AD ACCESSO DIRETTO

La creazione di un file ad accesso diretto richiede le seguenti fasi di programma:

1. Aprire il file per accesso diretto (con modalità "R"). Questo esempio specifica una lunghezza di record di 32 byte. Se la lunghezza del record viene omessa, il valore di default è pari a 128 byte.

   ```
   10 OPEN "R",#1,"FILE",32
   ```

2. Usare l'istruzione FIELD per allocare lo spazio nel buffer ad accesso diretto, per le variabili che verranno scritte sul file ad accesso diretto.

   ```
   20 FIELD#1,20 AS N$, 4 AS A$, 8 AS P$
   ```

3. Usare il comando LSET per trasferire i dati nel buffer ad accesso diretto. I valori numerici devono essere composti in stringhe quando vengono memorizzati nel buffer. Per fare

questo l'utente deve usare le funzioni di tipo "make".MKI$ trasforma un intero in una stringa, MKS$ converte un numero in singola precisione in una stringa e MKD$ converte un numero in doppia precisione in una stringa.

```
70 LSET N$=X$
80 LSET A$=MKS$(AMT)
90 LSET P$=TEL$
```

4. Trascrivere i dati dal buffer al disco tramite l'istruzione PUT.

```
100 PUT #1, CODE%
```

La funzione LOC, con i file ad accesso diretto, riporta il numero del record corrente, che è dato dal numero dell'ultimo record usato in un'istruzione GET o PUT piú uno. Per esempio l'istruzione:

```
200 IF LOC(1)>50 THEN END
```

termina l'esecuzione del programma se il numero di record corrente nel file #1 è superiore a 50.

Il programma 3 trasferisce le informazioni digitate su terminale al file ad accesso diretto.

```
10 OPEN "R",#1,"FILE",32
20 FIELD #1,20 AS N$, 4 AS A$, 8 AS P$
30 INPUT "2-DIGIT CODE";CODE%
40 INPUT "NAME";X$
50 INPUT "AMOUNT";AMT
60 INPUT "PHONE";TEL$:PRINT
70 LSET N$=X$
80 LSET A$=MKS$(AMT)
90 LSET P$=TEL$
100 PUT #1,CODE%
110 GOTO 30
```

Fig. 4-4 Programma 3 - Creazione di un File ad Accesso Diretto

Ogni volta che viene eseguita l'istruzione PUT, viene scritto un record sul file. Il codice a-due-cifre introdotto alla linea 30 diventa il numero di record.

Attenzione

Una variabile stringa definita tramite un'istruzione FIELD non deve essere usata in una istruzione INPUT o LET, perché, in questo caso, il puntatore a quella variabile si riferisce allo spazio stringa invece che al buffer del file ad accesso diretto.

COME ACCEDERE A UN FILE AD ACCESSO DIRETTO

La creazione di un file ad accesso diretto richiede le seguenti fasi:

1. Aprire il file con modalità "R"

   ```
   10 OPEN "R", #1, "FILE",32
   ```

2. Usare l'istruzione FIELD per allocare lo spazio nel buffer ad accesso diretto per le variabili che verranno lette dal file

   ```
   20 FIELD #1,20 AS N$, 4 AS A$, 8 AS P$
   ```

   Nota

   In un programma che esegue sia operazioni di input che di output sullo stesso file ad accesso diretto, spesso è possibile usare una sola istruzione OPEN ed una sola FIELD.

3. Usare l'istruzione GET per trasferire il record desiderato nel buffer ad accesso diretto

   ```
   40 GET #1,CODE%
   ```

4. Il programma può ora accedere ai dati contenuti nel buffer. I valori numerici che vengono letti da un buffer di file ad accesso diretto devono essere riconvertiti da stringhe in numeri

tramite l'uso di funzioni di tipo "convert". CVI converte una stringa di 2 byte in un numero intero, CVS converte una stringa di 4 byte in un numero in semplice precisione e CVD converte una stringa di 8 byte in un numero in doppia precisione.

```
50 PRINT N$

60 PRINT CVS(A$)
```

Il programma 4 permette di accedere al file ad accesso diretto "FILE" creato nel programma 3. Quando il codice a-due-cifre viene digitato sul terminale l'informazione associata a quel codice viene letta dal file e visualizzata.

```
10 OPEN "R",#1,"FILE",32
20 FIELD#1,20 AS N$, 4 AS A$, 8 AS P$
30 INPUT "2-DIGIT-CODE";CODE%
40 GET #1, CODE%
50 PRINT N$
60 PRINT USING "$$###.##";CVS(A$)
70 PRINT P$:PRINT
80 GOTO 30
```

Fig. 4-5 Programma 4 - Accesso a un File ad Accesso Diretto

Il programma 5 è un programma che illustra l'accesso ai file ad accesso diretto. In questo programma il numero di record viene usato come numero di matricola e si presuppone che l'inventario non contenga più di 100 numeri di matricola diversi. Le linee da 900 a 960 inizializzano il file dati scrivendo CHR$ (255) come primo carattere di ogni record. Questo viene utilizzato in seguito (linea 270 e linea 500) per stabilire se quel determinato numero di matricola è già stato introdotto.

Le linee da 140 a 210 visualizzano le diverse funzioni di inventario eseguite dal programma. Quando l'utente digita il numero di funzione desiderato la linea 230 permette di effettuare un salto al sottoprogramma appropriato.

```
120 OPEN"R",#,"INVEN.DAT",39
130 FIELD#1,1 AS F$,30 AS D$,2 AS Q$,2 AS R$,4 AS P$
140 PRINT:PRINT "FUNCTIONS:":PRINT
150 PRINT 1,"INITIALIZE FILE"
160 PRINT 2,"CREATE A NEW ENTRY"
170 PRINT 3,"DISPLAY INVENTORY FOR ONE PART"
180 PRINT 4,"ADD TO STOCK"
190 PRINT 5,"SUBTRACT FROM STOCK"
200 PRINT 6,"DISPLAY ALL ITEMS BELOW REORDER LEVEL"
210 PRINT:PRINT:INPUT"FUNCTION";FUNCTION
220 IF (FUNCTION<)OR(FUNCTION>6) THEN PRINT "BAD
    FUNCTION NUMBER":GOTO 140
230 ON FUNCTION GOSUB 900,250,390,480,560,680
240 GOTO 210
250 REM BUILD NEW ENTRY
260 GOSUB 840
270 IF ASC(F$)< >255 THEN INPUT"OVERWRITE";A$: IF
    A$< >"Y" THEN RETURN
280 LSET F$=CHR$(0)
290 INPUT "DESCRIPTION";DESC$
300 LSET D$=DESC$
310 INPUT"QUANTITY IN STOCK";Q%
320 LSET Q$=MKI$(Q%)
330 INPUT "REORDER LEVEL";R%
340 LSET R$=MKI$(R%)
350 INPUT "UNIT PRICE";P
360 LSET P$=MKS$(P)
370 PUT#1,PART%
380 RETURN
390 REM DISPLAY ENTRY
400 GOSUB 840
410 IF ASC(F$)=255 THEN PRINT "NULL ENTRY":RETURN
420 PRINT USING "PART NUMBER# # #";PART%
430 PRINT D$
440 PRINT USING "QUANTITY ON HAND# # # # #";CV1(Q$)
450 PRINT USING "REORDER LEVEL# # # # #";CVI(R$)
460 PRINT USING "UNIT PRICE $$# #.# #";CVS(P$)
470 RETURN
480 REM ADD TO STOCK
490 GOSUB 840
```

Fig. 4-6 Programma 5 - Inventario (cont.)

```
500 IF ASC(F$)=255 THEN PRINT "NULL ENTRY":RETURN
510 PRINT D$:INPUT"QUANTITY TO ADD";A%
520 Q%=CVI(Q$)+A%
530 LSET Q$=MKI$(Q%)
540 PUT#1,PART%
550 RETURN
560 REM REMOVE FROM STOCK
570 GOSUB 840
580 IF ASC(F$)=255 THEN PRINT"NULL ENTRY":RETURN
590 PRINT D$
600 INPUT "QUANTITY TO SUBTRACT";S%
610 Q%=CVI(Q$)
620 IF (Q%-S%)<0 THEN PRINT"ONLY";Q%;"IN STOCK":GOTO
    600
630 Q%=Q%-S%
640 IF Q%=<CVI(R$) THEN PRINT "QUANTITY NOW";Q%;"
    REORDER LEVEL";CVI(R$)
650 LSET Q$=MKI$(Q%)
660 PUT#1,PART%
670 RETURN
680 REM DISPLAY ITEMS BELOW REORDER LEVEL
690 FOR I,=1 TO 100
710 GET#1,I
720 IF CVI(Q$)<CVI(R$) THEN PRINT D$;"QUANTITY";CVI(Q$)
    TAB(50) "REORDER LEVEL";CVI(R$)
730 NEXT I
740 RETURN
840 INPUT "PART NUMBER";PART%
850 IF(PART%<1)OR(PART%>100) THEN PRINT "BAD PART
    NUMBER":GOTO 840 ELSE GET#1,PART%:RETURN
890 END
900 REM INITIALIZE FILE
910 INPUT "ARE YOU SURE";B$:IF B$<>"Y" THEN RETURN
920 LSET F$=CHR$(255)
930 FOR I=1 TO 100
940 PUT#1,I
950 NEXT I
960 RETURN
```

Fig. 4-6 Programma 5 - Inventario

PROTEZIONE DEL FILE

Ci sono tre istruzioni del GW-BASIC che permettono all'utente di controllare il corretto accesso ai file su disco: OPEN, LOCK e UNLOCK.

Le istruzioni LOCK e UNLOCK, e il parametro *access* dell'istruzione OPEN, possono solo essere usate se è stato installato MS-NET con MS-DOS 3.1 o future versioni.

Tutte e tre le istruzioni sono descritte nel Capitolo 8.

5. GRAFICA

SOMMARIO

L'interprete GW-BASIC, sotto MS-DOS, fornisce una gamma completa di prestazioni grafiche. È offerta una quantità notevole di possibilità per disegnare figure, usando vari colori. Il quadro video può essere suddiviso in più sezioni rettangolari, per visualizzare solo testo o testo e figure.

Questo capitolo introduce i concetti di base, i modi di grafica, l'uso dei colori, le coordinate, le istruzioni VIEW, WINDOW e altre.

INDICE

SELEZIONE DELL'AMBIENTE GRAFICO

Il linguaggio GW-BASIC fornisce un insieme di istruzioni e funzioni di grafica che consentono di visualizzare punti e linee su video usando vari colori. Per poter utilizzare queste istruzioni occorre selezionare l'ambiente grafico.

All'inizializzazione il sistema si trova in Modo Testo (SCREEN 0), è possibile entrare in ambiente grafico con l'istruzione SCREEN.

Ci sono tre diversi modi di grafica che possono essere selezionati:

- Modo a Media Risoluzione (tramite l'istruzione SCREEN 1)
- Modo ad Alta Risoluzione (tramite l'istruzione SCREEN 2)
- Modo ad Altissima Risoluzione (tramite l'istruzione SCREEN 3)

Si distinguono per il numero dei punti visualizzati e per il numero di colori permessi.

L'istruzione SCREEN pemette inoltre di selezionare la pagina su cui scrivere e quella da visualizzare in Modo Testo (con i parametri *apage* e *vpage*). Deve precedere ogni istruzione di I/O su video. Per default il sistema assume SCREEN 0,0,0,0; vengono selezionate 80 colonne in Modo Testo, e una pagina di video.

Possono essere usate piú istruzioni SCREEN per definire modalità video diverse per diverse sezioni di programma. È possibile inoltre passare da un modo di grafica ad un altro tramite l'istruzione WIDTH, che permette anche di definire l'ampiezza del video (in Modo Testo) e selezionare una finestra di testo.

MODO TESTO

Nel Modo Testo l'utente può visualizzare un testo, cioé lettere, numeri e tutti i caratteri speciali appartenenti all'insieme dei caratteri GW-BASIC. È possibile definire il colore di primo piano e di fondo per i caratteri tramite l'istruzione COLOR (Testo). Questa istruzione permette anche di creare caratteri lampeggianti, in negativo, invisibili, ad alta luminosità o sottolineati.

I caratteri vengono visualizzati in linee orizzontali dall'alto (linea 1) al basso (linea 25). Ogni linea ha 40 (o 80) colonne. Il comando WIDTH permette di selezionare il numero di colonne.

L'istruzione LOCATE posiziona il cursore su video. Le coordinate di colonna e di linea del cursore vengono calcolate dalle funzioni POS(0) e CSRLIN, rispettivamente

I caratteri vengono generalmente visualizzati, tramite l'istruzione PRINT o PRINT USING, alla posizione del cursore, da sinistra a destra su ciascuna linea, dalla linea 1 alla 24. Quando si tenta di posizionare il cursore sulla linea 25, le linee dalla 1 alla 24 scorrono di una linea verso l'alto.

La linea 25 viene di solito usata per visualizzare i valori associati ai tasti funzione.

Per muovere il cursore sulla linea 25 e visualizzare caratteri si usino le istruzioni KEY OFF, LOCATE e PRINT.

Sistema a più Pagine di Visualizzazione

Le pagine di visualizzazione multipla sono permesse in Modo Testo. Ogni istruzione che legge da o scrive su video (in Modo Testo) in realtà legge dalla o scrive sulla pagina attiva. La pagina di visualizzazione è quella che compare su video e può essere diversa dalla pagina attiva. Questa prestazione permette di visualizzare una pagina e contemporaneamente scrivere su un'altra. Le pagine attive e di visualizzazione possono essere selezionate dall'istruzione SCREEN.

Istruzioni e Funzioni

Le istruzioni e le funzioni disponibili nel Modo Testo per la visualizzazione di un testo sono:

ISTRUZIONI	FUNZIONI
CLS COLOR (Testo) LOCATE (Testo) PRINT PRINT USING SCREEN WIDTH WRITE	CSRLIN POS SCREEN SPC TAB

Nel Modo Testo è possibile selezionare i colori di primo piano e di fondo per i caratteri e farli lampeggiare.

In un sistema con video a colori sono disponibili 16 colori diversi, specificati da un codice:

0 nero	8 grigio
1 blu	9 blu chiaro
2 verde	10 verde chiaro
3 ciano	11 ciano chiaro
4 rosso	12 rosso chiaro
5 magenta	13 magenta chiaro
6 giallo	14 giallo chiaro
7 bianco	15 bianco ad alta intensità

In un sistema monocromatico sono disponibili soltanto due colori (nero e bianco), ma è possibile usare anche tonalità di grigio, caratteri sottolineati o ad alta intensità.

Per ulteriori dettagli si faccia riferimento all'istruzione COLOR (Testo) nel Capitolo 8.

MODI DI GRAFICA

In ciascuno dei tre modi di grafica è possibile, oltre che visualizzare un testo, tracciare delle figure complesse. Per visualizzare un testo sono utilizzabili tutte le istruzioni, i comandi e le funzioni

disponibili nel Modo Testo, fatta eccezione per le istruzioni COLOR (Testo) e LOCATE (Testo). Infatti nei modi di grafica l'utente deve usare le istruzioni COLOR (Grafica) e LOCATE (Grafica).

In grafica ogni punto del video è indirizzabile tramite le sue coordinate. Un punto sul video viene denominato 'pixel' (contrazione di "picture element") ed una linea di pixel viene denominata "scanline".

Istruzioni e Funzioni.

Le istruzioni e le funzioni che possono essere usate per la visualizzazione delle immagini (in ciascuno dei tre modi grafici) sono:

ISTRUZIONI	FUNZIONI
CIRCLE	PMAP
COLOR (Modo a Media Risoluzione)	POINT
COLOR (Modo ad Alta Risoluzione)	
COLOR (Modo ad Altissima Risoluzione)	
DRAW	
GET (Grafica)	
LINE	
LOCATE (Grafica)	
PAINT	
PRESET	
PSET	
PUT (Grafica)	
SCREEN	
VIEW	
WINDOW	

MODO A MEDIA RISOLUZIONE

In questo modo ci sono 320 pixel sull'asse orizzontale e 200 pixel su quello verticale. Questi sono numerati da sinistra a destra e dall'alto verso il basso, quindi il pixel nell'angolo in alto a sinistra è (0,0), mentre il pixel nell'angolo in basso a destra è (319, 199).

Se viene usato un video a colori, è possibile visualizzare fino a quattro colori contemporaneamente, con un video B/N i quattro colori appaiono come tonalità di grigio.

Come Tracciare le Figure

Quando si tracciano delle figure sul video per mezzo delle istruzioni di grafica (PSET, PRESET, LINE, CIRCLE, PAINT o DRAW) è possibile specificare un numero di colore pari a 0, 1, 2 o 3. Questo seleziona il colore dalla tavolozza corrente.

Se un'istruzione di grafica non specifica un numero di colore, viene assunto il colore di primo piano per la grafica, specificato tramite l'istruzione COLOR, oppure il valore 3 (se COLOR non specifica alcun colore di primo piano).

L'istruzione COLOR (a Media Risoluzione) permette di selezionare la tavolozza. Esistono due tavolozze, ciascuna di quattro colori, come riportato nella seguente tabella.

TAVOLOZZA	NUMERO DI COLORE			
	0	1	2	3
0 1	*background* *background*	verde ciano	rosso magenta	giallo bianco

Tab. 5-1 Tavolozze

Dove *background* è il primo parametro dell'istruzione COLOR; rappresenta un codice di colore ed è associato al numero di colore 0. Questo parametro specifica anche il colore di fondo per il testo.

Con un video monocromatico l'uso della memoria è identico. L'unica differenza consiste nel fatto che i due bit di un pixel vengono interpretati in modo diverso dall'hardware: visualizzando 4 tonalità di grigio.

Come Visualizzare i Caratteri

La dimensione dei caratteri nel Modo a Media Risoluzione è uguale a quella utilizzata nel Modo Testo, quando viene specificata un'ampiezza video di 40 colonne. Il colore di primo piano per i caratteri è stabilito dal parametro *tforeground* nell'istruzione COLOR (il valore di default è 3). Il colore di fondo per i caratteri è stabilito dal parametro *background* nell'istruzione COLOR (il valore di default è 0, cioé nero).

Se non viene usato il colore, il colore di primo piano per i caratteri è 1 (bianco) mentre il colore di fondo è 0 (nero).

MODO AD ALTA RISOLUZIONE

In questo modo ci sono 640 pixel sull'asse orizzontale e 200 pixel su quello verticale. Questi sono numerati da sinistra a destra e dall'alto verso il basso, quindi il pixel in alto a sinistra è (0,0) mentre quello in basso a destra è (639,199).

Ci sono solo due colori: nero (numero di colore 0) e bianco (numero di colore 1).

Come Tracciare le Figure

Quando si tracciano figure usando le istruzioni di grafica, è possibile specificare un numero di colore pari a 0, 1, 2 o 3.

Il colore 0 indica il nero, mentre il colore 1 indica il bianco. Il colore 2 è trattato come 0, il colore 3 come 1.

Se non si specifica un numero di colore il default risulta il colore di primo piano per la grafica, specificato dalla istruzione COLOR, o 1 (se COLOR non specifica alcun colore di primo piano).

L'istruzione COLOR permette di specificare i colori di primo piano e di fondo e, opzionalmente, un'operazione XOR tra i colori dei pixel su video ed i colori dei punti in memoria.

Come Visualizzare i Caratteri

La dimensione dei caratteri è uguale a quella utilizzata nel Modo Testo ad 80 colonne.

Il colore di primo piano per i caratteri è 1 (bianco) mentre il colore di fondo è 0 (nero).

MODO AD ALTISSIMA RISOLUZIONE

In questo modo ci sono 640 pixel sull'asse orizzontale e 400 pixel sull'asse verticale. Essi sono numerati da sinistra a destra e dall'alto verso il basso, quindi il pixel nell'angolo in alto a sinistra è (0,0) mentre il pixel nell'angolo in basso a destra è (639,399).

Ci sono solo due colori: nero (numero di colore 0) e bianco (numero di colore 1).

Come Tracciare le Figure

Quando si tracciano figure usando le istruzioni di grafica è possibile specificare un numero di colore pari a 0, 1, 2 o 3.

Il numero 0 indica il nero ed il numero 1 indica il bianco; il numero 2 è trattato come 0, ed il numero 3 come 1.

Se non viene specificato alcun numero di colore il default risulta essere il colore di primo piano per la grafica, specificato dall'istruzione COLOR, o 1 (se COLOR non specifica alcun colore di primo piano).

L'istruzione COLOR (Altissima Risoluzione) permette di specificare i colori di primo piano e di fondo per la grafica e, opzionalmente, un'operazione XOR tra i colori dei pixel su video ed i colori dei punti in memoria.

Tramite l'istruzione COLOR è possibile specificare la visualizzazione dei caratteri in negativo (reverse).

Come Visualizzare i Caratteri

La dimensione dei caratteri è uguale a quella utilizzata nel Modo Testo ad 80 colonne.

Il colore di primo piano per i caratteri è 1 (bianco) e quello di fondo è 0 (nero), a meno che venga specificato il video in negativo tramite l'istruzione COLOR.

FINESTRE

Nell'ambiente grafico GW-BASIC è possibile definire le seguenti finestre:

- Finestre logiche
- Finestre fisiche
- Finestre di testo

La finestra logica è una sezione rettangolare con dimensioni arbitrarie (non vincolate dalle dimensioni del video) su cui visualizzare la figura; viene definita tramite l'istruzione WINDOW.

La finestra fisica è una sezione rettangolare di video su cui viene proiettata la finestra logica associata, come riportato nella Figura 5-2.

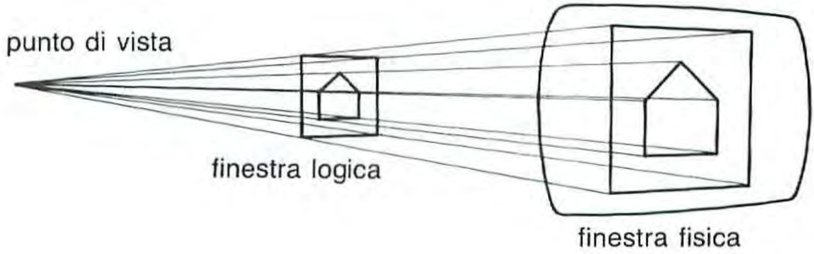


Fig. 5-2 Finestra Logica e Finestra Fisica

Il comando RUN e le istruzioni SCREEN e VIEW, senza parametri, definiscono una finestra fisica che coincide con l'intero video. L'istruzione VIEW con due coppie di coordinate fisiche definisce una finestra di area inferiore: la prima coppia individua il vertice superiore sinistro e la seconda il vertice inferiore destro della finestra.

È possibile definire piú finestre fisiche. Ad ogni esecuzione dell'istruzione VIEW viene definita una finestra che è quella corrente; per cambiare la finestra fisica corrente è necessario eseguire un'altra istruzione VIEW. Possono essere visualizzati solo i punti interni alla finestra fisica corrente; eventuali punti, riferiti in un'istruzione di grafica, esterni alla finestra fisica corrente non vengono visualizzati (effetto "clipping").

Oltre a definire una o più finestre fisiche è possibile definire una finestra di testo, cioé riservare un'area rettangolare su cui visualizzare il testo, tramite le istruzioni VIEW PRINT e WIDTH. Su quest'area è possibile scrivere o modificare un testo (utilizzando le prestazioni di Screen Editor) ma non visualizzare figure; inoltre la finestra di testo consente lo scorrimento automatico del testo quando viene completata l'ultima riga.

L'istruzione VIEW PRINT permette di definire una finestra di testo specificando la linea iniziale e quella finale; l'istruzione WIDTH permette di definire una finestra di testo specificando il numero di colonne a partire dal margine sinistro del video.

TRASFORMAZIONI WINDOW / VIEW

Variando l'area di una finestra, logica o fisica, la corrispondenza tra i punti delle due finestre cambia. Si possono ottenere quindi, dallo stesso disegno, immagini diverse rimpicciolendo, ingrandendo (effetto "zoom"), tagliando parte dell'immagine (effetto "clipping") oppure ingrandendo un particolare (effetti "zoom" e "clipping" combinati).

Variando l'area di una finestra ma mantenendo costante il rapporto tra i lati si ottengono immagini simili tra loro; variando invece questo rapporto si ottengono immagini distorte.

Definendo opportunamente più finestre fisiche parzialmente sovrapposte, è possibile comporre su video un disegno prelevandone i particolari da diversi disegni (cioé da diverse finestre logiche).

Un'ulteriore possibilità è quella di definire diverse finestre fisiche associate ad una stessa finestra logica, utilizzando una sola istruzione WINDOW e diverse istruzioni VIEW, tutte con l'opzione SCREEN. Questa opzione permette la proiezione dei punti della finestra logica sui punti dell'intero video, ma solo i punti all'interno della finestra fisica corrente sono visibili. In questo modo vengono visualizzati alcuni particolari del disegno.

Alcuni esempi delle possibilità descritte sono riportati per l'istruzione WINDOW, nel Capitolo 8.

COORDINATE

Ogni punto in una figura è individuato da coordinate, queste vengono interpretate dall'ambiente GW-BASIC come:

- coordinate fisiche o di device (in pixel)
- coordinate logiche o utente.

Le coordinate fisiche sono numeri interi positivi; l'origine degli assi è situata nel vertice in alto a sinistra del video, l'asse X è orientato verso destra e l'asse Y è orientato verso il basso.

Le coordinate logiche sono numeri reali con segno; l'origine e l'orientamento degli assi sono scelti dall'utente.

L'istruzione WINDOW permette di definire figure in uno spazio non delimitato dalle dimensioni fisiche del video, usando un sistema di coordinate cartesiane definito dall'utente.

Dopo aver eseguito un'istruzione WINDOW le coordinate di tutti i punti, specificati con istruzioni di grafica, sono interpretate da GW-BASIC come coordinate logiche; se non viene eseguita alcuna istruzione WINDOW le coordinate di tutti i punti specificati sono interpretate come coordinate fisiche. Fa eccezione l'istruzione DRAW che permette di specificare solo coordinate fisiche e disegna direttamente sulla finestra fisica.

Le coordinate logiche vengono convertite automaticamente da GW-BASIC in coordinate fisiche.

Se l'istruzione WINDOW viene eseguita senza parametri le coordinate logiche coincidono con le coordinate fisiche.

Indipendentemente dalla definizione di coordinate logiche o fisiche, le coordinate di un punto possono essere:

- coordinate assolute (nell'istruzione di grafica non è inclusa l'opzione STEP)
- coordinate relative (nell'istruzione di grafica è inclusa l'opzione STEP).

Le coordinate assolute indicano la posizione del punto rispetto all'origine del sistema di coordinate.

Le coordinate relative indicano la posizione del punto rispetto alle coordinate dell'ultimo punto a cui si è fatto riferimento.

L'esempio seguente illustra l'uso di entrambi i tipi di coordinate:

```
10 SCREEN 1 'Media Risoluzione
20 PSET (100, 50)
30 PSET STEP (10, -5)
```

vengono illuminati due pixel; il primo alle coordinate (100, 50) e il secondo alle coordinate (110, 45).

TECNICHE DI ANIMAZIONE

L'istruzione GET permette di memorizzare, negli elementi di un vettore numerico, i colori dei pixel compresi all'interno di un'area rettangolare su video. L'istruzione PUT consente poi di ritrasferire i colori su video, sulla stessa o su un'altra area.

È inoltre possibile, tramite l'istruzione PUT, eseguire operazioni logiche tra i colori memorizzati nel vettore e i colori dei pixel su video.

Per cancellare un'immagine su video è sufficiente ritrasferirla, mediante l'istruzione PUT, nella stessa posizione, specificando l'opzione XOR. Ciò consente l'animazione di oggetti.

Per spostare un oggetto su video in modo da farlo apparire in movimento è sufficiente effettuare i passi seguenti:

1. Visualizzare, usando l'istruzione PUT, la figura precedentemente memorizzata con GET

2. Calcolare la nuova posizione del rettangolo che la contiene

3. Trasferire, nella stessa posizione su video, la figura usando l'istruzione PUT con l'opzione XOR (in modo da cancellare l'oggetto senza modificare i colori di fondo)

4. Ritornare al punto 1 specificando, nell'istruzione PUT, la nuova posizione del rettangolo che contiene la figura

Per spostare più figure è necessario ripetere questa procedura per ciascun oggetto.

6. PRESTAZIONI SPECIALI

SOMMARIO

Questo capitolo è destinato agli utenti che intendono richiamare sottoprogrammi in linguaggio macchina da GW-BASIC o usare rilevamento di eventi o processi "child".

In particolare si parla di: allocazione di memoria per sottoprogrammi in linguaggio macchina, caricamento in memoria di sottoprogrammi, chiamate di sottoprogrammi da GW-BASIC e passaggio dei parametri, rilevamento di eventi e processi "child".

Tale capitolo è destinato a programmatori esperti.

INDICE

SOTTOPROGRAMMI IN LINGUAGGIO MACCHINA

I sottoprogrammi in linguaggio macchina possono essere richiamati da un programma GW-BASIC tramite la funzione USR o le istruzioni CALL o CALLS.

La funzione USR permette di richiamare un sottoprogramma in linguaggio macchina, che restituisce un valore, nello stesso modo in cui vengono richiamate le funzioni intrinseche GW-BASIC. Si raccomanda, comunque, di usare l'istruzione CALL o CALLS per richiamare sottoprogrammi in linguaggio macchina da GW-BASIC. Queste istruzioni consentono il passaggio di piú argomenti. Inoltre l'istruzione CALL è compatibile con un numero maggiore di versioni BASIC che non la funzione USR.

ALLOCAZIONE DI MEMORIA

Prima di caricare in memoria un sottoprogramma in linguaggio macchina è necessario allocare uno spazio di memoria sufficiente. Per fare ciò esistono due possibilità.

La prima consiste nell'allocare spazio per i sottoprogrammi in linguaggio macchina, all'interno del segmento dati GW-BASIC, utilizzando l'istruzione CLEAR oppure l'opzione /M: del comando GWBASIC. Per i sottoprogrammi in linguaggio macchina possono venire riservate solo le locazioni piú alte di memoria del segmento dati GW-BASIC.

Se è richiesto uno spazio maggiore di stack, quando viene richiamato un programma in linguaggio macchina, è possibile salvare lo stack del GW-BASIC e preparare un nuovo stack che viene messo a disposizione del sottoprogramma in linguaggio macchina. Lo stack di GW-BASIC deve essere ripristinato prima del rientro da sottoprogramma.

La seconda possibilità consiste nell'allocare spazio per i sottoprogrammi in linguaggio macchina all'esterno del segmento dati GW-BASIC, utilizzando l'istruzione DEF SEG con l'indirizzo di inizio segmento opportuno. Questo indirizzo deve essere più alto dell'indirizzo di fine del segmento dati GW-BASIC (ottenuto sommando 64K all'indirizzo di inizio del segmento dati).

La scelta tra le due possibilità illustrate dipende essenzialmente dalla memoria disponibile.

CARICAMENTO IN MEMORIA DI SOTTOPROGRAMMI

Un sottoprogramma in linguaggio macchina può essere caricato in memoria con diversi procedimenti, il più semplice dei quali è costituito dal comando BLOAD. È anche possibile usare il comando SHELL per caricare ed eseguire un programma che rimane residente dopo essere stato eseguito. Il terzo procedimento consiste nell'eseguire un programma che rimane residente ed eseguire poi GW-BASIC.

È necessario osservare le seguenti direttive se si sceglie di usare il comando BLOAD per caricare in memoria un file EXE o per leggerlo e scriverlo:

1. Assicurarsi che i sottoprogrammi non contengano riferimenti lunghi e spiazzamenti che superino 64K o che portino fuori dal segmento di codice. I riferimenti lunghi devono essere gestiti tramite il caricatore EXE.

2. Saltare i primi 512 byte del file di output del linker (EXE), leggendo quindi il resto del file.

Le due sezioni che seguono illustrano due modi standard per caricare in memoria sottoprogrammi in linguaggio macchina.

CARICAMENTO TRAMITE L'ISTRUZIONE POKE

I sottoprogrammi in linguaggio macchina possono essere caricati in memoria tramite l'istruzione POKE nel modo seguente: dopo la fase di assemblaggio del sottoprogramma è necessario creare delle istruzioni DATA contenenti il valore, in esadecimale, di ogni byte di codice (rappresentato come &Hxx). Il sottoprogramma viene quindi scritto (tramite l'istruzione POKE) nell'area di memoria specificata, byte per byte.

Il sottoprogramma può poi essere richiamato tramite la funzione USR o l'istruzione CALL. Se viene usata la funzione USR deve essere specificato l'indirizzo di ingresso nel sottoprogramma, tramite un'istruzione DEF USR. In questo modo viene definita la posizione di chiamata, nell'ambito del segmento corrente, della funzione definita tramite USR. Se viene usata l'istruzione CALL, l'indirizzo di ingresso nel sottoprogramma è costituito dal valore della variabile numerica specificata subito dopo CALL. Questa variabile deve contenere lo spiazzamento rispetto all'origine del segmento corrente. In entrambi i casi il segmento viene definito tramite l'istruzione DEF SEG.

CARICAMENTO TRAMITE IL COMANDO BLOAD

Un sottoprogramma in linguaggio macchina può essere caricato in memoria tramite il comando BLOAD nel modo seguente:

1. Creare, per prima cosa, un file .EXE del sottoprogramma usando il linker e caricare poi l'interprete GW-BASIC sotto DEBUG digitando:

   DEBUG GWBASIC.EXE

2. Visualizzare e registrare i valori contenuti nei registri CS, IP, SS, SP, DS ed ES, usando il comando R (per usarli al punto 5), per determinare la locazione di GW-BASIC in memoria.

3. Caricare il file .EXE ad un indirizzo di memoria alto, usando DEBUG, sovrapponendolo alla sezione transiente di COMMAND.COM.

4. Visualizzare i registri, usando il comando R, per determinare la locazione di memoria del sottoprogramma. I valori dei registri CS ed IP devono essere memorizzati per essere usati ai punti 6 e 7.

5. Assegnare ai registri i loro valori originali memorizzati al punto 2, usando il comando R. Nel sottoprogramma possono venire inseriti, opzionalmente, dei punti di interruzione, usando il comando G che salta al punto di ingresso di GW-BASIC.

6. Caricare il programma applicativo. Modificare l'istruzione DEF SEG insieme all'istruzione DEF USR o alla variabile dell'istruzione CALL, in modo da farle corrispondere alla locazione di memoria del sottoprogramma, come specificato al punto 4 (cioé il valore del registro IP per DEF USR o per la variabile dell'istruzione CALL).

7. L'area di memoria del sottoprogramma deve essere salvata (tramite BSAVE) nel modo diretto GW-BASIC, usando entrambi i valori dei registri CS ed IP a partire dal punto 4, insieme al listing in linguaggio assembler o alla lunghezza del codice mappa di LINK.

8. Assicurarsi che il programma applicativo contenga un'istruzione DEF SEG, con l'appropriato valore del registro CS, seguita da un'istruzione BLOAD. BLOAD può caricare un sottoprogramma in una locazione alternativa, se il sottoprogramma è autorilocabile. In questo caso, ricordarsi anche di modificare l'istruzione DEF SEG associata.

9. Infine, deve essere registrato su disco il programma applicativo aggiornato.

Nota

Se GW-BASIC viene eseguito sotto DEBUG, DEBUG viene caricato in memoria per primo, per evitare che venga cancellato e riscritto. Eventuali punti di interruzione, o il comando SYSTEM, restituiscono il controllo a DEBUG.

COME RICHIAMARE IL SOTTOPROGRAMMA DA GW-BASIC

ISTRUZIONE CALL

L'istruzione CALL rappresenta il modo più comune per richiamare programmi in linguaggio macchina da GW-BASIC. Essa è preferibile alla funzione USR (introdotta quasi esclusivamente per problemi di compatibilità con i programmi che ne fanno uso).

La sintassi di un'istruzione CALL è:

CALL *numvar* [(*variable* [,*variable*]...)]

Dove

numvar contiene lo spiazzamento, all'interno del segmento corrente, del punto di partenza del sottoprogramma che deve essere richiamato.

variable è una variabile che deve essere passata al sottoprogramma come argomento.

Il segmento corrente è quello di default (segmento dati GW-BASIC) o quello che è stato specificato dall'ultima istruzione DEF SEG eseguita.

Il richiamo dell'istruzione CALL provoca i seguenti eventi:

1. Per ogni variabile specificata nell'istruzione, lo spiazzamento (su due byte) della locazione della variabile all'interno del segmento dati GW-BASIC viene memorizzato nello stack.

2. L'indirizzo di ritorno specificato nel registro CS e lo spiazzamento vengono memorizzati nello stack.

3. Viene trasferito il controllo alla routine in linguaggio macchina tramite l'indirizzo di segmento, dato nell'ultima istruzione DEF SEG eseguita, e lo spiazzamento fornito da *numvar*.

I seguenti schemi illustrano rispettivamente lo stato dello stack, quando viene attivata l'istruzione CALL durante l'esecuzione del sottoprogramma richiamato.

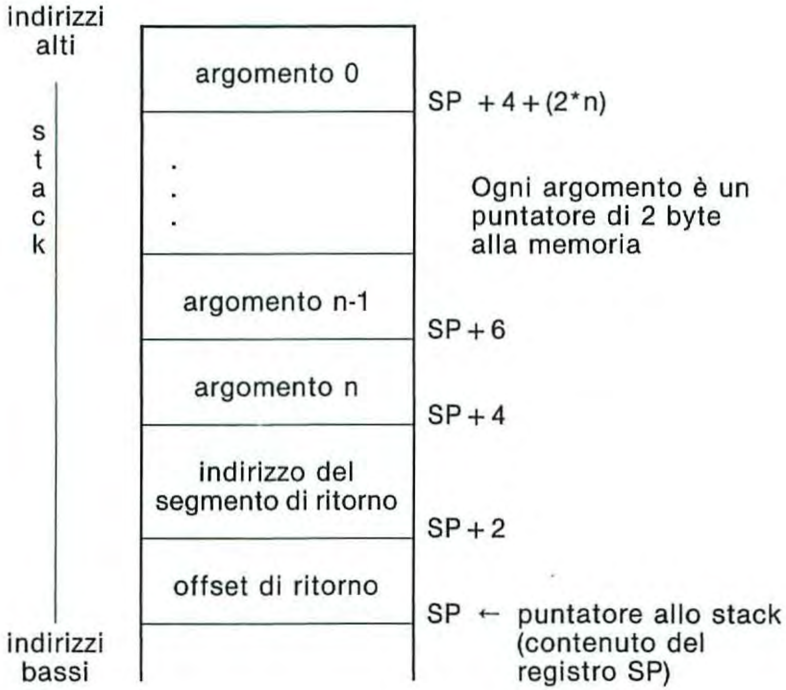


Fig. 6-1 Schema dello Stack quando Viene Attivata l'Istruzione CALL.

Dopo che l'istruzione CALL è stata attivata, il controllo viene trasferito al sottoprogramma. È possibile indirizzare gli argomenti caricando il puntatore base (BP) con il valore del puntatore allo stack (SP), aggiungendo a quest'ultimo uno spiazzamento positivo.



Fig. 6-2 Schema dello Stack durante l'Esecuzione di una Istruzione CALL

Durante la codifica di un sottoprogramma è necessario osservare le seguenti regole:

1. La routine richiamata deve proteggere i registri di segmento DS, ED, SS e BP. Se durante l'escuzione del sottoprogramma le interruzioni vengono disabilitate, esse devono essere abilitate prima di ritornare al programma principale. Prima del ritorno al programma chiamante, lo stack deve essere correttamente riposizionato.

2. Il sottoprogramma chiamato deve conoscere il numero e la lunghezza degli argomenti passati. La routine che segue illustra un modo semplice di fare riferimento agli argomenti:

   ```
   push    BP
   mov     BP,SP
   add     BP, (2* numero degli argomenti) + 4
   ```

   Allora:

   l'argomento 0 è a BP
   l'argomento 1 è a BP-2
   l'argomento n è a BP-2*n

   (numero degli argomenti = n + 1)

3. Le variabili possono essere allocate o nel Segmento Codice o sullo stack. Attenzione a non modificare il segmento di ritorno e lo spiazzamento memorizzati nello stack.

4. Il sottoprogramma chiamato deve azzerare lo stack. Il modo migliore per fare ciò consiste nell'eseguire il rientro al programma principale con RET *n* (dove *n* è due volte il numero di argomenti presenti nella lista). Ciò riporta lo stack alla condizione precedente la sequenza di chiamata.

5. I valori vengono restituiti all'interprete GW-BASIC includendo nella lista degli argomenti il nome della variabile che deve contenere il risultato.

6. Se l'argomento è una stringa, lo spiazzamento dell'argomento punta a 3 byte che, presi come unità, vengono chiamati "descrittore di stringa". Il byte 0 del descrittore di stringa contiene la lunghezza della stringa (da 0 a 255). I byte 1 e 2 sono, rispettivamente, gli 8 bit piú bassi e più alti dell'indirizzo di partenza della stringa nello spazio stringhe.

   Nota

   Se l'argomento è una costante stringa, il descrittore di stringa punta al testo del programma. Attenzione a non alterare o distruggere in questo modo il programma. Per evitare risultati imprevedibili, è necessario aggiungere + "" alla costante stringa definita nel programma. Usare per esempio:

   ```
   20 A$ = "BASIC"+""
   ```

   Questo fa sì che la costante stringa venga copiata nello spazio stringhe. La stringa può quindi essere modificata senza variare il programma.

7. Il contenuto di una stringa può essere modificato tramite routine utente, ma il descrittore non deve essere cambiato. Non si deve scrivere oltre la fine della stringa. GW-BASIC non può manipolare correttamente le stringhe se la loro lunghezza viene modificata da routine esterne.

8. Le aree dati necessarie per la routine devono essere allocate o nel segmento CODE della routine utente o sullo stack. Non è possibile dichiarare un'area dati separata nella routine utente in linguaggio macchina.

Esempio

```
100 DEF SEG=&H8000
110 VAR=&H7FA
120 CALL VAR (A, B$, C)
 .
 .
 .
```

La linea 100 posiziona il segmento a 80000 Hex. Il valore della variabile VAR viene aggiunto all'indirizzo poiché l'ultima word dopo il valore DEF SEG viene spostata a sinistra di 4 bit, cioé moltiplicata per 16 (questa è una funzione del microprocessore, non di GW-BASIC). In questo caso la variabile VAR è inizializzata a &H7FA in modo da far eseguire, alla chiamata a VAR, il sottoprogramma allocato all'indirizzo 80000:7FA Hex (indirizzo assoluto 8007FA Hex).

Il seguente esempio in linguaggio assembler 8086 illustra il modo di accesso agli argomenti passati. Il risultato restituito viene memorizzato nella variabile 'C'.

```
PUSH    BP:            ;Posiziona il puntatore agli
                       argomenti
MOV     BP,SP
ADD     BP,(4+2*3)
MOV     BX,[BP-2]      ;Preleva l'indirizzo del descrittore
                       di B$
MOV     CL,[BX[        ;Preleva la lunghezza di B$ e la
                       trasferisce in CL.
MOV     D,1[BX]        ;Preleva l'indirizzo del testo di B$
                       e lo trasferisce in DX.
        .
        .
        .
MOV     SI,[BP]        ;Preleva l'indirizzo di 'A' e lo
                       trasferisce in SI.
MOV     DI[BP-4]       ;Preleva il puntatore a 'C' e lo
                       trasferisce in DI.
MOVS    WORD           ;Memorizza la variabile 'A' in 'C'.
POP     BP             ;Riposiziona il puntatore.
RET     6              ;Riposiziona lo stack e ritorna.
```

Nota

Il sottoprogramma chiamato deve conoscere il tipo di variabile degli argomenti numerici. Nell'esempio precedente l'istruzione:

MOVS WORD

copia solamente due byte. Questo funziona se le variabili A e C sono intere. Se invece le variabili sono in semplice precisione si devono copiare 4 byte e nel caso di variabili in doppia precisione si devono copiare 8 byte.

ISTRUZIONE CALLS

L'istruzione CALLS deve essere usata per accedere a sottoprogrammi scritti usando le convenzioni di chiamata MS-FORTRAN. L'istruzione CALLS agisce esattamente come CALL, ma gli argomenti vengono passati come indirizzi segmentati, invece che come indirizzi non segmentati.

Poiché le routine MS-FORTRAN devono conoscere il valore del segmento per ogni argomento passato, il segmento viene memorizzato nello stack con lo spiazzamento. Vengono utilizzati quattro byte per ogni argomento, invece di 2, come nell'istruzione CALL; perciò se una routine scritta in linguaggio assembler viene attivata tramite l'istruzione CALLS, nell'istruzione RET, *n* è quattro volte il numero degli argomenti.

FUNZIONE USR

Come si è detto, il modo più comune per richiamare routine in linguaggio macchina è rappresentato dall'uso dell'istruzione CALL, tuttavia è anche possibile usare a questo scopo la funzione USR. In questo modo viene assicurata la compatibilità con i programmi contenenti funzioni USR.

Il formato della funzione USR è:

USR[*n*] (*argument*)

Dove

n	è una cifra da 0 a 9, che specifica quale routine USR deve essere richiamata. Se *n* viene omesso si assume USR0.
argument	rappresenta qualsiasi espressione numerica o stringa, passata come argomento al sottoprogramma. Questo parametro deve essere sempre specificato anche se non è richiesto dal sottoprogramma (impostando un argomento fittizio).

Per ogni funzione USR deve essere eseguita la funzione DEF USR corrispondente, per definire lo spiazzamento rispetto all'indirizzo iniziale del segmento corrente. Questo spiazzamento e l'indirizzo specificato dall'ultima DEF SEG eseguita determinano l'indirizzo di partenza del sottoprogramma.

Quando viene effettuata la chiamata della funzione USR, il registro AL contiene un valore che specifica il tipo di argomento. Il valore di AL può essere:

2	intero su due byte (complemento a due)
3	stringa
4	numero a virgola mobile in semplice precisione
8	numero a virgola mobile in doppia precisione.

Se l'argomento è un numero, il registro BX punta all'Accumulatore di Virgola Mobile (FAC) dove è memorizzato l'argomento. FAC è un'area di 8 byte nello spazio dati GW-BASIC.

Se l'argomento è un intero:

FAC-2 contiene gli 8 bit più significativi dell'argomento
FAC-3 contiene gli 8 bit meno significativi dell'argomento.

Se l'argomento è un numero a virgola mobile in semplice precisione (24 bit di mantissa):

FAC-2 contiene gli 8 bit medi della mantissa
FAC-3 contiene gli 8 bit più bassi della mantissa.

Se l'argomento è un numero a virgola mobile in doppia precisione (56 bit di mantissa): da FAC-4 a FAC-7 sono contenuti i 4 ulteriori byte della mantissa (FAC-7 contiene gli 8 bit più bassi).

Se l'argomento è una stringa, il registro DX punta a 3 byte che, presi come unità, vengono chiamati "descrittore di stringa". Il byte 0 del descrittore di stringa contiene la lunghezza della stringa (da 0 a 255 caratteri). I byte 1 e 2 sono, rispettivamente, gli 8 bit inferiori e superiori dell'indirizzo di partenza della stringa nell'ambito del segmento dati GW-BASIC. Se l'argomento è una costante stringa definita nel programma, il descrittore di stringa punta al testo del programma. Attenzione a non alterare o distruggere, in questo caso, il programma.

Generalmente, il valore restituito da una funzione USR è dello stesso tipo (intero, stringa, semplice precisione o doppia precisione) di quello dell'argomento passato.

GW-BASIC ha esteso l'interfaccia della funzione USR per permettere di richiamare le routine MAKINT e FRCINT. In questo modo è possibile accedere a queste routine senza specificare i loro indirizzi assoluti. L'indirizzo ES:BP viene usato come puntatore remoto indiretto alle routine FRCINT e MAKINT.

Per richiamare FRCINT tramite una routine USR si deve usare CALL DWORD ES:[BP].

Per richiamare MAKINT tramite una routine USR si deve usare CALL DWORD ES:[BP+4].

Esempio

```
110 DEF USR0=&H8000 'Si assume che l'utente abbia
                     specificato /M:32767
120 X=5
130 Y=USR0 (X)
140 PRINT Y
```

Il tipo della variabile (numerica o stringa) che riceve la chiamata di funzione deve essere conforme a quello dell'argomento passato.

RILEVAMENTO DI EVENTI

Il rilevamento di eventi permette di trasferire il controllo ad una linea di programma quando si verifica un determinato evento. Il controllo viene trasferito, come se fosse stata eseguita un'istruzione GOSUB, alla "routine di rilevamento", che inizia al numero di linea specificato. Dopo la gestione dell'evento, la routine di rilevamento esegue l'istruzione RETURN che fa riprendere l'esecuzione del programma dal punto in cui essa si trovava quando si è verificato l'evento.

Gli eventi che vengono rilevati possono essere la ricezione di caratteri da una porta di comunicazione (ON COM(*n*) GOSUB), la pressiome di determinati tasti (ON KEY(*n*) GOSUB), lo scadere di un intervallo di tempo (ON TIMER(*n*) GOSUB) oppure lo svuotamento della coda di background per la musica (ON PLAY(*n*) GOSUB).

Il rilevamento di eventi è controllato dalle seguenti istruzioni:

Sintassi 1 (per attivare il rilevamento)

{COM(*n*) I KEY(*n*) I TIMER I PLAY} ON

Sintassi 2 (per disattivare il rilevamento)

{COM(*n*) I KEY(*n*) ITIMER I PLAY} OFF

Sintassi 3 (per disattivare temporaneamente il rilevamento)

{**COM**(*n*) | **KEY**(*n*) | **TIMER** | **PLAY**} **STOP**

Osservazioni

COM(*n*) — dove *n* è il numero della porta di comunicazione.

In genere, prima di ritornare, la routine di rilevamento dell'evento COM legge un intero messaggio dalla porta specificata. Non si consiglia di gestire come evento la ricezione di ogni singolo carattere. In questo caso si potrebbe verificare l'overflow del buffer COM.

KEY(*n*) — dove *n* è il numero del tasto la cui pressione deve essere segnalata. I tasti di questo tipo sono numerati da 1 a 20.

Si noti che l'istruzione KEY(*n*) ON non è uguale a KEY ON. L'istruzione KEY(*n*) ON imposta il rilevamento di eventi per il tasto specificato. L'istruzione KEY ON visualizza i valori di tutti i tasti funzione sulla linea 25 del video.

Quando GW-BASIC è in modo diretto, i tasti funzione mantengono i loro significati standard.

Quando viene rilevata la pressione di un tasto l'occorrenza di quel tasto viene distrutta. Quindi non si possono utilizzare le istruzioni INPUT o INKEY$ per conoscere il tasto che ha causato l'evento. Così, se si desidera assegnare funzioni diverse a determinati tasti, è necessario impostare un sottoprogramma per ogni tasto, piuttosto che assegnare le diverse funzioni in un unico sottoprogramma.

TIMER — L'istruzione ON TIMER(*n*) GOSUB (dove *n* è un'espressione numerica che rappresenta il numero di secondi a partire dalla mezzanotte precedente) può essere usata per l'esecuzione di una data routine allo scadere di un intervallo di tempo.

PLAY — L'istruzione ON PLAY(*n*) GOSUB (dove *n* è un numero di note lasciate nel buffer di musica) viene usata per estrarre più note dalla coda di background per la musica, così da avere una musica di sottofondo, durante l'esecuzione del programma.

ISTRUZIONE ON GOSUB

L'istruzione ON GOSUB stabilisce un numero di linea per il rilevamento di eventi. Il suo formato è:

ON {COM(*n*) | KEY(*n*) | TIMER(*n*) | PLAY(*n*) }GOSUB *linenum*

---

linenum uguale a zero disabilita il rilevamento di quell'evento.

Quando un evento è attivo (ON) e se nell'istruzione ON GOSUB è stato specificato un numero di linea diverso da zero, ogni volta che GW-BASIC inizia una nuova istruzione controllerà che l'evento specificato sia accaduto (per esempio se è entrato un carattere COM). Quando un evento è disattivato (OFF) il suo rilevamento non avviene, l'evento non viene memorizzato anche se si verifica.

Quando un evento è bloccato (STOP) non viene effettuato il suo rilevamento, l'evento viene memorizzato in modo da permettere immediatamente il suo rilevamento dopo l'esecuzione dell'istruzione ON associata.

Quando viene effettuato il rilevamento di un particolare evento viene provocato automaticamente uno stop su tale evento, in questo modo viene evitata la situazione ricorsiva. Un rientro dalla routine di rilevamento esegue automaticamente l'istruzione ON, a meno che all'interno della routine di rilevamento non sia stata eseguita un'istruzione OFF.

Occorre notare che dopo il rilevamento di errore, la funzione di rilevamento è automaticamente disabilitata. Inoltre, non si verificherà mai un rilevamento di eventi se GW-BASIC non esegue un programma.

ISTRUZIONE RETURN

Quando è in corso un rilevamento di eventi, non appena si verifica l'evento specificato viene eseguita l'istruzione GOSUB. Per esempio:

ON KEY(10) GOSUB 1000

specifica che il programma continua alla linea 1000 non appena il tasto funzione F10 viene premuto. Se al termine di questo sottoprogramma viene eseguita una semplice istruzione RETURN, il controllo del programma ritorna all'istruzione seguente quella in cui è avvenuto il rilevamento. Quando l'istruzione RETURN è stata eseguita, il corrispondente salto all'indirizzo di ritorno viene cancellato dallo stack.

GW-BASIC include l'istruzione RETURN *linenum*, che permette al processo di elaborazione di riprendere da una linea ben definita. In genere, quando si incontra un'istruzione RETURN il programma ritorna all'istruzione che segue l'istruzione GOSUB. Comunque RETURN *linenum* permette di specificare un'altra linea. Se non viene usata con attenzione questa prestazione può dare origine a problemi. Supponiamo, per esempio, che il programma contenga:

```
10 ON KEY(10) GOSUB 1000
20 FOR I = 1 TO 10
30 PRINT I
40 NEXT I
50 REM LINEA DI PROGRAMMA SEGUENTE

200 'QUI RICOMINCIA IL PROGRAMMA

1000 'PRIMA LINEA DEL SOTTOPROGRAMMA
.
.
.
1050 RETURN 200
```

Se viene premuto il tasto funzione F10 durante l'esecuzione del ciclo FOR/NEXT viene eseguito il sottoprogramma ma, invece di completare il ciclo FOR/NEXT, viene restituito il controllo alla linea 200. L'ingresso originale di GOSUB viene quindi cancellato dall'istruzione RETURN e qualsiasi altra istruzione GOSUB, WHILE o FOR, che era attiva al momento del rilevamento, rimane attiva. Anche il contesto corrente di FOR rimane attivo, e può verificarsi l'errore "FOR without NEXT".

GW-BASIC E PROCESSI "CHILD"

Tramite l'uso del comando SHELL, GW-BASIC è in grado di utilizzare una delle caratteristiche rilevanti di MS-DOS: la possibilità di creare processi "child". Il comando SHELL permette di elaborare una parte del programma GW-BASIC, di uscire temporaneamente in MS-DOS, per eseguire una funzione specificata, e di ritornare al programma GW-BASIC nel punto in cui si trova l'istruzione che segue il comando SHELL per continuare con il programma.

GW-BASIC crea un programma "child" quando viene usato il comando SHELL. È impossibile per GW-BASIC riuscire a proteggersi completamente dai suoi processi "child". Quando è stato eseguito un comando SHELL, possono verificarsi molte cose; per esempio, i file possono essere aperti (OPEN) ed i device possono essere in funzione.

Le seguenti indicazioni aiutano ad evitare un danno all'ambiente GW-BASIC da parte dei processi "child".

HARDWARE

In genere, durante un comando SHELL si raccomanda di preservare lo stato di tutto l'hardware. L'interfaccia di implementazione rende possibile questa raccomandazione. Può comunque risultare indispensabile l'astensione dall'uso di determinati device, all'interno dei processi "child", che vengono eseguiti tramite il comando SHELL di GW-BASIC. Le aree specifiche sono le seguenti:

1. Device Video. Mentre un processo "child" può visualizzare un'informazione utile, altri processi "child" possono danneggiare i parametri già esistenti.

2. Vettori di Interruzione. Salvano e ripristinano i vettori di interruzione che il processo "child" ha intenzione di usare.

3. Altri hardware. Molti device vengono sistemati da GW-BASIC in uno stato specifico. In tali device possono essere compresi Controllori di Interruzione, Counter Timer, controllore di DMA, I/O Latch e Uarts. Questi device possono essere utilizzati dal processo "child" senza alcuna limitazione.

IL FILE SYSTEM

Un processo "child" che altera un qualsiasi file aperto precedentemente in GW-BASIC può portare a risultati disastrosi.

Se tali file devono essere aggiornati è necessario chiuderli (CLOSE) prima di effettuare il comando SHELL, e quindi riaprirli subito al rientro in GW-BASIC (vedere "Reindirizzamento di Input ed Output Standard" al Capitolo 8, nel comando GWBASIC).

GESTIONE DELLA MEMORIA

1. Prima di trasmettere un comando SHELL a COMMAND, GW-BASIC cerca di liberare lo spazio di memoria che non usa tranne in un caso: quando viene usato lo switch /M: nel comando GWBASIC. In questo caso il sistema assume che l'utente abbia intenzione di introdurre routine in testa al segmento riservato all'area di lavoro GW-BASIC. Questo serve ad evitare che venga ridotta l'area di lavoro prima di eseguire il comando SHELL. Per questo motivo quando viene usato lo switch /M:, il comando SHELL può generare l'errore "Out of memory".

È preferibile caricare i sottoprogrammi in linguaggio macchina prima di attivare GW-BASIC. Questo può essere realizzato con l'inserimento di "codice pocket" alla fine dei sottoprogrammi in linguaggio macchina, per permettere a loro di uscire in MS-DOS e rimanere residenti. Per esempio:

```
CSEG          SEGMENT CODE
              .
              .
              .
; Sottoprogramma in linguaggio macchina
              .
              .
              .
              RET        ; Ultima istruzione
START::
              INT 27 H;   Terminato, rimane residente
CSEG          ENDS
              END         START
```

per essere sicuri di caricare questi programmi prima di GW-BASIC conviene eseguirli. A questo scopo risulta utile il file AUTOEXEC.BAT.

2. Un processo "child" non deve mai terminare e rimanere residente, poiché in questo modo GW-BASIC può non avere abbastanza spazio per espandere il suo spazio lavoro fino alla dimensione originale. Se GW-BASIC non può ripristinare lo spazio lavoro tutti i file vengono chiusi, viene stampato il messaggio che comunica l'errore "SHELL can't continue" e da GW-BASIC si esce in MS-DOS.

3. Nella porzione di GW-BASIC indipendente dalla macchina non c'é alcun vincolo che proibisca a GW-BASIC di essere eseguito come un processo "child" di se stesso. Comunque date le complicazioni che possono sorgere da questa struttura, è preferibile non fare uso di questa prestazione.

7. COMUNICAZIONI ASINCRONE

SOMMARIO

In questo capitolo viene descritto come usare l'interprete GW-BASIC per supportare comunicazioni asincrone RS232 con unità periferiche ed altri computer.

È destinato a programmatori che hanno una conoscenza di base delle tecniche di programmazione.

INDICE

COME APRIRE I FILE DI COMUNICAZIONE

L'istruzione OPEN COM alloca un buffer per l'input e l'output in modo simile all'istruzione OPEN per file su disco. Per una descrizione completa si faccia riferimento al Capitolo 8.

I/O DI COMUNICAZIONE

Dato che la porta di comunicazione viene aperta come un file, tutte le istruzioni Input/Output valide per file su disco sono anche valide per COM.

Le istruzioni di input sequenziale COM sono uguali a quelle per file su disco e sono INPUT#, LINE INPUT# e la funzione INPUT$.

Le istruzioni di output sequenziale COM sono uguali a quelle per disco e sono PRINT#, PRINT#USING e WRITE#.

Per dettagli sulla sintassi e sull'uso della codifica occorre consultare le istruzioni INPUT e PRINT nel Capitolo 8.

Le istruzioni GET e PUT sono leggermente diverse per i file COM. Si faccia riferimento alle istruzioni GET (File COM) e PUT (File COM) descritte nel Capitolo 8.

FUNZIONI I/O DI COMUNICAZIONE

La principale difficoltà della comunicazione asincrona è costituita dalla necessità di elaborare i caratteri alla stessa velocità in cui vengono ricevuti. A velocità superiori ai 2400 bps. può essre necessario sospendere la trasmissione dal computer host in modo da non perdere nessuno dei caratteri che si stanno ricevendo. Si può fare ciò inviando XOFF (CHR$(19)) all'host e poi XON (CHR$(17)) per far riprendere la trasmissione una volta pronti a ricevere.

GW-BASIC fornisce le funzioni che aiutano a capire l'imminenza di una condizione di "over-run":

LOC(*f*) Riporta il numero dei caratteri presenti nel buffer di input in attesa di essere letti. Il buffer di input può contenere più di 255 caratteri (determinati dall'opzione /C:). Se nel buffer sono presenti più di 255 caratteri, LOC(*f*) riporta 255. Poiché una stringa è limitata a 255 caratteri, questo limite pratico significa che non è necessario controllare la lunghezza della stringa prima di leggere i dati in essa contenuti. Se nel buffer rimangono meno di 255 caratteri, LOC(*f*) ne riporta il numero effettivo.

LOF(*f*) Riporta la quantità di spazio libero rimasta nel buffer di input, cioé la dimensione del buffer LOC(*f*), dove dimensione rappresenta la misura del buffer di comunicazione definita dall'opzione /C:. LOF può essere usato per rivelare quando il buffer di input sta raggiungendo la sua capacità massima.

EOF(*f*) Riporta "vero" (-1), se il buffer di input è vuoto, e "falso" (0), se ci sono caratteri in attesa di essere letti.

Errori Possibili

"Communication buffer overflow": viene fatto un tentativo di leggere quando il buffer di input è pieno (cioé LOF(*f*) riporta 0).

"Device I/O error": durante la ricezione viene riscontrata una delle seguenti condizioni di linea: Overrun Error (OE), Framing Error (FE) o Break Interrupt (BI). L'errore viene eliminato dagli input successivi, ma il carattere che ha causato l'errore viene perso.

"Device fault": il Data Set Ready (DSR) viene perso durante una operazione di I/O.

FUNZIONE INPUT$ PER FILE DI COMUNICAZIONE

Durante la lettura di file COM, la funzione INPUT$ è preferibile alle istruzione INPUT# e LINE INPUT#, poiché tutti i caratteri ASCII possono essere significativi nella comunicazione. L'istruzione INPUT# è meno indicata, perché l'input viene interrotto quando vengono ricevuti una virgola (,) o **CR** e l'istruzione LINE INPUT# termina quando viene ricevuto **CR**.

INPUT$ permette a tutti i caratteri letti di essere assegnati ad una stringa. Si ricordi che INPUT$ (*n*, *f*) restituisce *n* caratteri letti dal file #*f*. Le seguenti istruzioni sono quindi le più efficienti per leggere un file COM:

```
10 WHILE NOT EOF(1)
20  A$=INPUT$(LOC(1),#1)
30 ...
40 ... Elaborazione dati restituiti in A$
50 ...
60 WEND
```

Le istruzioni sopra riportate restituiscono i caratteri, eventualmente presenti nel buffer, in A$ e li elaborano. Se sono presenti più di 255 caratteri, ne vengono letti solo 255 alla volta, per evitare overflow di stringa. Se ciò si verifica, EOF (1) è falso e l'input continua finché il buffer di input non è vuoto. La sequenza di eventi è quindi semplice, concisa e veloce.

ESEMPIO DI UN PROGRAMMA DI COMUNICAZIONE

Il programma che segue permette di usare il Personal Computer come un terminale convenzionale. Oltre alla comunicazione Full Duplex con un altro computer, il programma TTY permette ai dati ricevuti di essere trasferiti su un file. Viceversa, un file su disco può essere trasferito in memoria e trasmesso ad un altro computer.

Oltre a descrivere gli elementi di una comunicazione asincrona, questo programma è utile durante il trasferimento di programmi GW-BASIC e di dati al sistema e dal sistema.

```
10 REM
20 REM *** Programma di Test RS232
30 REM
40 SCREEN 0,0
50 KEY OFF:CLS:CLOSE
60 DEFINT A-Z
62 LOCATE 25,1
64 PRINT STRING$(60," ")
70 FALSE=0:TRUE=NOT FALSE
80 MENU = 5
90 XOFF$ = CHR$(19):XON$=CHR$(17)
92 LOCATE 25,1: PRINT "TTY Async Program"
94 LOCATE 1,1: LINE INUT "Baud?"; SPEED$
96 LOCATE 25,1: PRINT "TTY Async Program"
100 ON COM(1) GOSUB 730
110 COMFIL$="COM1:"+SPEED$+",E,7"
120 OPEN COMFIL$ AS 1
130 REM
140 REM *** Modo Talk ***
150 REM
160 CLS
170 LOCATE 25,1:PRINT"RS232 test program running in TALK
    MODE";
180 PAUSE=FALSE
190 LOCATE 1,1
200 A$=INKEY$:IF A$=""THEN 220
210 IF ASC(A$)=MENU THEN 290 ELSE PRINT #1,A$;
220 IF EOF(1) THEN 200
230 IF LOC(1)>50 THEN PAUSE=TRUE: PRINT #1, XOFF$
240 A$=INPUT$(LOC(1),1)
250 PRINT A$;:IF LOC(1)>0 THEN 230
260 IF PAUSE THEN PAUSE=FALSE: PRINT #1, XON$;
270 GOTO 200
280 REM
290 REM *** Modo Command ***
300 REM
```

Fig. 7-1 Programma TTY (cont.)

```
310 CLS
320 LOCATE 25,1:PRINT"RS232 test program running in
    COMMAND MODE";
330 LOCATE 1,1
340 INPUT"FILE":DSKFIL$
350 LOCATE 1,1:PRINT STRING$ (80," "):LOCATE 1,1
360 INPUT "(T)ransmit or (R)eceive ":TXRX$
370 IF TXRX$="T" THEN OPEN DSKFIL$ FOR
    INPUT AS 3:GOTO 580
380 IF TXRX$="R" THEN 410
390 GOTO 350
400 REM
410 REM *** Modo Receive ***
420 REM
430 LOCATE 25,32 : PRINT "FILE RECEIVE MODE";
440 OPEN DISKFIL$ FOR OUTPUT AS 3
450 IF EOF(1): THEN GOSUB 520
460 IF LOC(1)>50 THEN PAUSE=TRUE: PRINT #1,XOFF$;
470 A$=INPUT$(LOC(1),1)
480 PRINT #3,A$;
490 IF LOC(1)>0 THEN 460
500 IF PAUSE THEN PAUSE=FALSE:PRINT #1, XON$;
510 GOTO 450
520 FOR I=1 TO 5000
530 IF NOT EOF(1) THEN RETURN
540 NEXT I
550 CLOSE #3
560 RETURN 140
570 REM
580 REM *** Modo Transmit ***
590 REM
600 LOCATE 25,32:PRINT"FILE TRANSMIT MODE";
610 COM(1) ON
620 XFLAG=1
630 WHILE NOT EOF(3)
640 A$=INPUT$(1,3)
650 WHILE XFLAG=0 :WEND
660 PRINT #1,(A$);
670 WEND
680 COM(1) OFF
```

Fig. 7-1 Programma TTY (cont.)

```
690 PRINT #1,CHR$(26);
700 CLOSE 3
710 GOTO 140
720 REM
730 REM *** Routine di ricezione XON/XOFF ***
740 REM
750 IF EOF(1) THEN RETURN
760 B$=INPUT$(LOC(1),1)
770 IF LEN(B$)=2 THEN 790
780 IF B$=XOFF$ THEN 810
790 XFLAG=1
800 RETURN
810 XFLAG=0
820 RETURN
```

Fig. 7-1 Programma TTY

Note sull'Esempio di Programmazione TTY

Numero di Linea	Commento
10-90	Definisce gli attributi del video ed inizializza le variabili del programma.
92-96	Consente di introdurre da tastiera la velocità di comunicazione (in bps).
100	Specifica il numero di linea della prima istruzione della routine di rilevamento di eventi di comunicazione associata al canale numero 1.
110-120	Apre ed inizializza il canale di comunicazione 1 con la velocità introdotta da tastiera (in bps), parità pari e 7 bit dati.
170	Visualizza sulla 25ª linea del video un messaggio indicante il modo operativo (Modo Talk).

180-260	Invia i caratteri impostati da tastiera al canale numero 1 e visualizza i caratteri ricevuti dal canale. L'istruzione 210 trasferisce il controllo all'istruzione 290 (dove si entra nel Modo Command) se l'utente imposta **CTRL E**.
320	Visualizza sulla 25ª linea del video un messaggio che indica il nuovo modo (Modo Command).
340	Chiede all'utente di impostare il nome del file da trasmettere o da ricevere, a seconda del carattere (T o R) introdotto durante l'esecuzione dell'istruzione 360.
370	Se l'utente introduce T, apre il file specificato per l'istruzione INPUT e salta all'istruzione 580.
380	Se l'utente introduce R, salta all'istruzione 410 (dove si entra in Modo Receive).
410-560	Il programma è in Modo Receive come visualizzato dall'istruzione 430 sulla 25ª linea del video. L'istruzione 440 apre il file specificato per l'istruzione OUTPUT. L'istruzione 450 controlla che non ci siano caratteri in attesa sul buffer di ricezione. Se non vi sono caratteri in attesa, il controllo viene trasferito all'istruzione 520, altrimenti all'istruzione seguente (linea 460).
520-540	Viene attivato un ciclo FOR/NEXT in attesa dell'arrivo di caratteri nel buffer di ricezione. Se non arriva nessun carattere entro il numero di iterazioni specificato, si esce dal Modo Receive. Il controllo viene quindi trasferito all'istruzione 550, dove il file viene chiuso e poi all'istruzione 140, ritornando in Modo Talk. Se invece arrivano dei caratteri, il controllo viene trasferito all'istruzione 460.
460	Controlla se il numero di caratteri presenti nel buffer di ricezione è maggiore di 50. In questo caso viene inviato un carattere XOFF$ al canale, per interrompere la trasmissione.

470-480	Se il numero di caratteri presenti nel buffer di ricezione è inferiore o uguale a 50, i caratteri vengono letti dal buffer di ricezione e scritti sul file.
490-510	Controlla se vi sono ancora dei caratteri sul buffer di ricezione. Nel caso affermativo, viene trasferito il controllo all'istruzione 460. Nel caso negativo, viene inviato un carattere XON$ (se era stato inviato in precedenza un carattere XOFF$) e il controllo viene trasferito all'istruzione 450.
580-710	Il programma è in Modo Transmit, come visualizzato dall'istruzione 600 sulla 25ª linea del video. Il file è già stato aperto per l'input all'istruzione 370. L'istruzione 610 abilita il rilevamento di eventi di comunicazione. Le istruzioni da 630 a 670 formano un ciclo WHILE/WEND che legge e trasmette il file (l'istruzione 640 legge un carattere alla volta e l'istruzione 660 lo invia al canale di comunicazione). La trasmissione dei caratteri viene sospesa se viene ricevuto un carattere XOFF$ (istruzione 650). La trasmissione dei caratteri viene ripresa se viene ricevuto un carattere XON$. Le istruzioni dalla 680 alla 710 disabilitano il rilevamento di eventi di comunicazione, inviano un carattere EOF, chiudono il file e ritornano in Modo Talk.
730-820	Formano la routine di rilevamento di eventi di comunicazione. L'istruzione 750 controlla se vi sono dei caratteri in attesa nel buffer di ricezione. Se nessun carattere è in attesa, viene eseguita un'istruzione RETURN. Se due caratteri sono in attesa viene abilitata la trasmissione dei caratteri (istruzione 790) e si esce dalla routine (RETURN). La presenza di due caratteri nel buffer di ricezione significa che sono stati ricevuti sia un carattere XON$ sia un carattere XOFF$. Se un solo carattere è in attesa la trasmissione dei caratteri viene disabilitata (se questo carattere è XOFF$) oppure abilitata (se questo carattere è XON$).

PARTE II

8. COMANDI, ISTRUZIONI E FUNZIONI

SOMMARIO

In questo capitolo vengono descritti, in modo dettagliato, i comandi, le istruzioni e le funzioni disponibili in GW-BASIC, con i loro esempi.

INDICE

INTRODUZIONE

Questo capitolo contiene le informazioni relative ai comandi, alle istruzioni e alle funzioni usate in GW-BASIC. Le prime pagine del capitolo elencano i comandi, le istruzioni e le funzioni, raggruppandole in classi funzionali. Segue un elenco completo, in ordine alfabetico, in cui vengono illustrati, per ogni comando, istruzione o funzione, lo scopo, la sintassi, le caratteristiche, uno o più esempi ed eventuali osservazioni.

Istruzioni e Comandi

Risulta a volte difficile distinguere un'istruzione GW-BASIC da un comando GW-BASIC, dal momento che entrambi possono essere usati in una linea di programma o in una linea immediata.

Le istruzioni GW-BASIC vengono generalmente usate in linee di programma e impostate in sequenza per formare un programma (ad eccezione di PRINT, LPRINT, LET che possono comparire anche in linee immediate, quando il sistema viene usato come un calcolatore o per operazioni di debugging).

I comandi GW-BASIC vengono generalmente usati nelle linee immediate (ad eccezione di SYSTEM, TROFF e TRON spesso usati anche in un programma) per manipolare programmi e per svolgere attività di servizio (ad esempio listare programmi o azzerare la memoria).

Funzioni

Una funzione può essere considerata un algoritmo che dà un unico risultato e può essere di tipo utente o di sistema (built-in). Essa può essere richiamata digitando semplicemente il suo nome, seguito da uno o più argomenti (tra parentesi), che rappresentano i valori che i parametri della funzione devono assumere. Una funzione può essere richiamata sia da una linea di programma sia da una linea immediata (ad eccezione di EOF che può solo essere richiamata da una linea di programma).

Una funzione può essere di tipo numerico, se restituisce un valore numerico, o di tipo stringa, se restituisce un valore di tipo stringa.

In questo capitolo vengono elencate tutte le funzioni di sistema (built-in). Per informazioni su come calcolare funzioni matematiche, che possono essere facilmente derivate da funzioni di sistema, occorre consultare l'Appendice B.

COMANDI

Quello che segue è un elenco di tutti i comandi usati in GW-BASIC.

AUTO

BLOAD

BSAVE

CHDIR

CLEAR

CONT

DELETE

EDIT

FILES

GWBASIC (è un comando MS-DOS, non un comando GW-BASIC)

KILL

LCOPY

LIST

LLIST

LOAD

MERGE

MKDIR

NAME

NEW

RENUM

RESET

RMDIR

RUN

SAVE

SHELL

SYSTEM

TRON

TROFF

ISTRUZIONI

Questa parte elenca tutte le istruzioni GW-BASIC, suddivise in due categorie separate: Istruzioni non di I/O ed Istruzioni di I/O.

ISTRUZIONI NON DI I/O

CALL

CALLS

CHAIN

COM(*n*)

COMMON

DATE$

DEF FN

DEF SEG

DEF USR

DEFINT, DEFSNG, DEFDBL e DEFSTR

DIM

END

ENVIRON

ERASE

ERROR

FOR/NEXT

GOSUB/RETURN

GOTO

IF ... GOTO ... ELSE e IF ... THEN ... ELSE

KEY

KEY(*n*)

LET

MID$

ON COM(*n*) GOSUB

ON ERROR GOTO

ON ... GOSUB

ON ... GOTO

ON KEY(*n*) GOSUB

ON PLAY(*n*) GOSUB

ON TIMER(*n*) GOSUB

OPTION BASE

PLAY

PLAY { ON I OFF I STOP }

POKE

RANDOMIZE

REM

RESTORE

RESUME

STOP

SWAP

TIME$

TIMER

WAIT

WHILE...WEND

ISTRUZIONI DI I/O

BEEP

CIRCLE

CLOSE

CLS

COLOR (Modo Testo)

COLOR (Grafica a Media Risoluzione)

COLOR (Grafica ad Alta Risoluzione)

COLOR (Grafica ad Altissima Risoluzione)

DATA

DRAW

FIELD

GET (File)

GET (File COM)

GET (Grafica)

INPUT#

INPUT

IOCTL

LINE

LINE INPUT

LINE INPUT#

LOCATE (Testo)

LOCATE (Grafica)

LPRINT

LPRINT USING

LSET/RSET

OPEN

OPEN COM

OUT

PAINT

PRESET

PRINT

PRINT USING

PRINT#

PRINT#USING

PSET

PUT (File)

PUT (File COM)

PUT (Grafica)

READ

SCREEN

SOUND

VIEW

VIEW PRINT

WIDTH

WINDOW

WRITE

WRITE#

FUNZIONI

In questa sezione vengono elencate tutte le funzioni "built-in" (o intrinseche) usate in GW-BASIC. Esse vengono suddivise in due categorie: funzioni di tipo numerico e di tipo stringa. Ulteriori sottoclassi vengono definite all'interno di ogni categoria.

FUNZIONI NUMERICHE (restituiscono un valore numerico)

Funzioni Aritmetiche

ABS

ATN

CDBL

CINT

COS

CSNG

EXP

FIX

INT

LOG

RND

SGN

SIN

SQR

TAN

Funzioni Numeriche su Stringhe

ASC

{ CVI I CVS I CVD }

INSTR

LEN

VAL

Funzioni di I/O e Varie

CSRLIN

EOF

ERDEV

ERL

ERR

FRE

INP

LOC

LOF

LPOS

PEEK

PLAY(*n*)

PMAP

POINT

POS

SCREEN

TIMER

USR

VARPTR

FUNZIONI STRINGA (restituiscono un valore stringa)

Funzioni Generali

CHR$

LEFT$

MID$

RIGHT$

SPACE$

STRING$

Funzioni di I/O e Varie

DATE$

ENVIRON$

ERDEV$

HEX$

INKEY$

INPUT$

IOCTL$

{ MKI$ I MKS$ I MKD$ }

OCT$

SPC

STR$

TAB

TIME$

VARPTR$

ABS Funzione

Calcola il valore assoluto di un'espressione numerica.

ABS(*numexp*)

Caratteristiche

Il valore assoluto di un'espressione numerica è sempre maggiore o uguale a zero.

Esempio 1

```
Ok
PRINT ABS (8*(-6))
 48
Ok
```

Esempio 2

```
20 DISTANCE = ABS(START-FINISH)
```

Esempio 3

```
100 IF ABS(DELTA)< =LIMIT THEN STOP
```

ASC Funzione

Restituisce il codice ASCII del primo carattere di una stringa data.

ASC(*stringexp*)

Caratteristiche

La funzione ASC restituisce il codice ASCII (0 - 255) corrispondente al primo carattere della stringa *stringexp*. L'elenco completo di tutti i codici ASCII è riportato nell'Appendice C.

Se il valore dell'argomento è la stringa nulla, si verifica l'errore "Illegal function call".

Si veda la funzione CHR$ per la conversione da codice decimale ASCII a stringa.

Esempio

Nell'esempio seguente si può vedere che il codice ASCII per la lettera maiuscola 'T' è 84.

```
10 X$ = "TEST"
20 PRINT ASC(X$)
RUN
 84
Ok
```

ATN Funzione

Calcola l'arcotangente dell'argomento.

ATN(*numexp*)

Caratteristiche

Il calcolo di ATN viene eseguito in precisione semplice, se non viene specificato /D nella linea di comando GWBASIC.

Il risultato viene espresso in radianti e si colloca nell'intervallo da -PI/2 a PI/2 (dove PI = 3.141593).

Esempio 1

```
10 INPUT X
20 PRINT ATN(X)
RUN
? 3
 1.249046
Ok
```

Esempio 2

```
100 IF ATN(N)<PI/2.0 THEN PRINT "ANGLE < 90 DEGREES"
```

AUTO Comando

Genera un numero di linea dopo ogni **CR**. Questo comando può essere eseguito solo in modo immediato.

AUTO [*linenum*][,[*increment*]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
linenum	È il numero usato per iniziare la numerazione delle linee.
increment	È il valore aggiunto a un numero di linea per ottenere il numero di linea successivo.

Caratteristiche

Il comando AUTO inizia la numerazione da *linenum* e ogni numero di linea successivo viene ottenuto sommando a *linenum increment*. Il default per entrambi i valori è 10. Se *linenum* è seguito da una virgola e *increment* non è specificato, viene utilizzato l'ultimo incremento specificato in un comando AUTO. Se non è stato impostato, in precedenza, nessun comando AUTO viene utilizzato un incremento pari a 10.

Se AUTO genera un numero di linea già utilizzato, dopo tale numero viene visualizzato un asterisco per avvertire l'utente che sta per sostituire una linea già esistente. Digitanto **CR** subito dopo

l'asterisco viene lasciata inalterata la linea esistente e viene generato il numero di linea successivo.

Per terminare la numerazione automatica si prema **CTRL BREAK** o **CTRL C**; la linea attuale viene cancellata e GW-BASIC ritorna in stato comandi.

Esempi

AUTO

Genera i numeri di linea 10, 20, 30, 40...

AUTO 100,20

Genera i numeri di linea 100, 120, 140...

AUTO 200,

Genera i numeri di linea 200, 220, 240, 260...
L'incremento è 20 perché questo è l'incremento definito nel comando AUTO precedente.

AUTO ,15

Genera i numeri di linea 0, 15, 30, 45...

BEEP Istruzione

Attiva il segnalatore acustico.

BEEP

Caratteristiche

L'istruzione PRINT CHR$(7) invia un carattere ASCII BEL che provoca lo stesso effetto dell'istruzione BEEP.

Esempio

```
130 IF X < 20 THEN BEEP
```

BLOAD Comando

Carica in memoria una copia del file.

BLOAD {*filespec*|*pathname*} [,*offset*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filespec (o *pathname*)	È un'espressione stringa che specifica il file che deve essere caricato. Se il device viene omesso, viene preso in considerazione il drive di default di MS-DOS.
offset	È un'espressione intera compresa nell'intervallo da 0 a 65535. Indica lo spiazzamento all'interno del segmento dichiarato, per mezzo dell'ultima istruzione DEF SEG da cui deve iniziare il caricamento.

Caratteristiche

I comandi BLOAD e BSAVE permettono di caricare in memoria e registrare su un file routine scritte in linguaggio macchina. Quando queste routine sono residenti in memoria, possono essere richiamate dal programma GW-BASIC tramite un'istruzione CALL.

I comandi BLOAD e BSAVE permettono anche di caricare e registrare qualsiasi parte di memoria; per esempio, è possibile registrare e visualizzare immagini video (specificando il buffer del video come segmento corrente tramite un'istruzione DEF SEG).

Se *offset* viene omesso, viene utilizzato lo spiazzamento specificato da BSAVE e il file viene caricato a partire dalla stessa locazione di memoria dalla quale è stato registrato.

Se *offset* è specificato, un'istruzione DEF SEG deve essere eseguita prima di BLOAD. Quando viene specificato *offset* GW-BASIC assume che l'utente voglia caricare a partire da un indirizzo diverso da quello di salvataggio; viene usato l'ultimo indirizzo di DEF SEG. Se non è stata usata nessuna istruzione DEF SEG in precedenza, viene usato come default il Segmento Dati GW-BASIC (poiché esso rappresenta il default per DEF SEG),

Attenzione

BLOAD non controlla l'estensione dell'indirizzo, cioé è possibile caricare un file in qualsiasi punto della memoria. L'utente deve avere cura di non sovrapporre il file al codice GW-BASIC o al sistema operativo.

Esempio 1

```
10 'Carica un programma in linguaggio macchina in memoria
   alla locazione 60:F000
20 DEF SEG   'Ripristina il segmento al DS del GW-BASIC.
30 BLOAD "B:PROG1", &HF000   'Carica PROG1 nel DS.
```

Esempio 2

```
10 'Carica il buffer di video
20 DEF SEG = &HB800 'Punta il segmento al buffer di video
30 BLOAD "FILE1",0 'Carica FILE1 nel buffer di video
```

Si noti l'istruzione DEF SEG alla linea 20 e lo spiazzamento 0 alla linea 30: ciò garantisce che venga usato l'indirizzo corretto.

Un esempio riportato nel comando seguente (BSAVE) illustra il modo in cui è stato registrato FILE1.

BSAVE Comando

Registra sezioni della memoria principale nel file specificato.

BSAVE [*filespec*|*pathname*],*offset*,*length*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filespec (*o pathname*)	È un'espressione stringa che specifica il nome del file che deve essere registrato e, opzionalmente, un device. Se il device viene omesso, viene utilizzato il drive di default di MS-DOS.
offset	È un'espressione intera compresa tra 0 e 65535. Indica lo spiazzamento all'interno del segmento dichiarato dall'ultima istruzione DEF SEG.

ELEMENTO DI SINTASSI	SIGNIFICATO
length	È un'espressione intera compresa tra 1 e 65535 che indica la lunghezza della sezione di memoria che deve essere registrata.

Caratteristiche

Un file immagine della memoria è una copia byte per byte del contenuto della memoria.

I comandi BLOAD e BSAVE permettono di caricare in memoria e registrare routine scritte in linguaggio macchina. Quando queste routine sono residenti in memoria, possono essere richiamate dal programma GW-BASIC tramite un'istruzione CALL.

I comandi BLOAD e BSAVE permettono anche di caricare in memoria e registrare qualsiasi parte di memoria; per esempio, è possibile registrare e visualizzare immagini video (specificando il buffer del video come segmento corrente tramite un'istruzione DEF SEG).

Un'istruzione DEF SEG deve essere eseguita prima di BSAVE. Per la registrazione viene sempre usato l'ultimo indirizzo di DEF SEG conosciuto.

Esempio 1

```
10 'Registra PROG1
20 DEF SEG = &H6000
30 BSAVE "PROG1",&HF000,256
```

Questo esempio registra 256 byte, a partire dalla locazione 60000:F000, nel file PROG1.

Esempio 2

```
10 'Registra il buffer di video
20 DEF SEG = &HB800 'Punta il segmento al buffer di video
30 BSAVE "A:FILE1",0,16384 'Registra il buffer di video nel file
                            FILE1
```

L'istruzione DEF SEG deve essere usata per posizionare l'indirizzo di segmento all'inizio del buffer di video. Spiazzamento uguale a zero e lunghezza uguale a 16384 indicano che deve essere registrato tutto il buffer di video (16K).

CALL Istruzione

Trasferisce il controllo a un sottoprogramma scritto in linguaggio macchina. CALL viene generalmente usata all'interno di un programma.

CALL *numvar* [(*variable*[,*variable*]...)]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
numvar	È una variabile numerica contenente l'indirizzo di memoria da cui inizia il sottoprogramma. Tale indirizzo è uno spiazzamento nel segmento di memoria corrente.

ELEMENTO DI SINTASSI	SIGNIFICATO
variable	È una variabile numerica o stringa che viene passata come argomento al sottoprogramma.

Caratteristiche

L'istruzione CALL è un modo per trasferire il controllo del programma a un sottoprogramma esterno. Un altro modo è rappresentato dalla fuzione USR.

Esempio

```
110 MYROUT = &HD000
120 CALL MYROUT (I,J,K)
        .
        .
        .
```

CALLS Istruzione

L'istruzione CALLS è uguale all'istruzione CALL con le eccezioni che vengono citate in seguito.

CALLS *numvar* **[(***variable***[,***variable***]...)]**

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
numvar	È una variabile numerica contenente l'indirizzo di memoria da cui inizia il sottoprogramma.
variable	È una variabile numerica o stringa che viene passata come argomento al sottoprogramma.

Caratteristiche

L'istruzione CALLS è simile a CALL, con un'eccezione: vengono passati gli indirizzi segmentati di tutti gli argomenti (CALL passa gli indirizzi non segmentati). L'istruzione CALLS deve essere usata per accedere a routine scritte con le convenzioni di chiamata FORTRAN, poiché tutti i parametri FORTRAN sono indirizzi segmentati richiamabili tramite riferimento.

L'istruzione CALLS usa l'indirizzo del segmento definito dall'ultima istruzione DEF SEG eseguita per allocare la routine chiamata.

CDBL Funzione

Converte un'espressione numerica in un numero in doppia precisione.

CDBL(*numexp***)**

Esempio

```
10 A=454.67
20 PRINT A; CDBL(A)
RUN
 454.67 454.6700134277344
Ok
```

CHAIN Istruzione

Trasferisce il controllo e passa le variabili a un altro programma. L'istruzione CHAIN viene usata solamente all'interno di un programma.

CHAIN [**MERGE**] {*filespec*|*pathname*} [,[*linenum*]
[,[**ALL**] [,**DELETE** *range*]]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filespec (o *pathname*)	È un'espressione stringa che specifica il nome del file di programma richiamato e, opzionalmente, il drive. Se il drive viene omesso, viene preso in considerazione il drive di default di MS-DOS.
linenum	Può essere un numero di linea o una espressione il cui valore rappresenta un numero di linea del programma richiamato. Questo parametro costituisce il punto di partenza per l'esecuzione del programma richiamato. Se questo parametro viene omesso, l'esecuzione del programma inizia alla prima linea. Il comando RENUM non modifica il valore di questo parametro.
range	Rappresenta il numero di linee che specificano la sezione dei programma corrente da cancellare, prima di registrare il programma concatenato (vedere il secondo esempio, piú avanti). Questi numeri possono venire modificati dal comando RENUM.

Caratteristiche

SE...	ALLORA...
viene usata l'opzione MERGE	viene eseguita un'operazione di MERGE tra il programma attualmente in memoria e il programma concatenato. Quest'ultimo deve essere stato registrato su disco in formato ASCII. Se il file su disco contiene delle linee con gli stessi numeri di quelle del programma in memoria, queste ultime vengono sostituite dalla prime. Un'operazione di MERGE può essere considerata come un inserimento delle linee del programma su disco nel programma presente in memoria. L'opzione MERGE lascia i file aperti, conserva il posizionamento corrente di OPTION BASE, i tipi di variabile e le funzioni definite dall'utente per essere utilizzate dal programma concatenato. Le funzioni definite dall'utente devono essere inserite nel programma prima di istruzioni CHAIN MERGE, altrimenti le funzioni rimangono non definite dopo che l'operazione di MERGE è stata ultimata.
viene omessa l'opzione MERGE	il programma attualmente in memoria viene perso (ad eccezione delle variabili comuni) prima di caricare in memoria il programma concatenato. L'istruzione CHAIN non conserva tipi di variabile o funzioni utente. Perciò tutte le istruzioni DEF *type* o DEF FN contenenti variabili comuni devono essere ripetute nel programma concatenato.

▶

SE...	ALLORA...
viene usata l'opzione ALL	i valori di tutte le variabili del programma attualmente in memoria vengono trasferiti al programma concatenato.
viene usata l'opzione ALL e viene omesso *linenum*	devono essere inserite due virgole tra *filespec* (o *pathname*) e l'opzione ALL. 100 CHAIN "NEXTPROG",,ALL è corretto; 100 CHAIN "NEXTPROG",ALL è errato. In questo caso GW-BASIC considera ALL un nome di variabile e lo valuta come un numero di linea.
viene omessa l'opzione ALL	il programma attualmente in memoria deve contenere una o più istruzioni COMMON per listare le variabili che vengono trasferite.
viene usata l'opzione DELETE	una sezione del programma attualmente in memoria viene cancellata prima di caricare in memoria il programma concatenato. L'opzione DELETE viene spesso usata con le opzioni MERGE e *linenum* per caricare in memoria degli overlay. Dopo che un overlay è stato caricato ed eseguito, è consigliabile cancellarlo per permettere di caricare ed eseguire un altro overlay.

Osservazioni

L'istruzione CHAIN compie un'operazione di RESTORE prima di eseguire un programma concatenato. In questo modo il puntatore viene riposizionato all'inizio del file dati interno.

Esempio 1

L'istruzione CHAIN viene usata in modi diversi nei due programmi che seguono. Nel primo, i due vettori stringa vengono dimensionati e dichiarati come variabili comuni. Quando il primo programma giunge alla linea 80, viene concatenato al secondo programma (PROG2), che carica i due elementi del vettore B$. Alla linea 80 di PROG2, il controllo ritorna al primo programma concatenato, iniziando l'esecuzione alla linea 90. Questo processo può essere osservato attraverso il testo descritto che viene visualizzato durante l'esecuzione dei programmi.

```
10 REM QUESTO PROGRAMMA MOSTRA L'USO DI CHAIN
   UTILIZZANDO VARIABILI COMMON
20 REM SALVA QUESTO MODULO SU DISCO COME "PROG1"
   USANDO L'OPZIONE A
30 DIM A$(2),B$(2)
40 COMMON A$( ),B$( )
50 A$(1) = "VARIABLES IN COMMON MUST BE ASSIGNED"
60 A$(2) = "VALUES BEFORE CHAINING."
70 B$(1) = "": B$(2) = ""
80 CHAIN "PROG2"
90 PRINT: PRINT B$(1): PRINT: PRINT B$(2): PRINT
100 END
```

```
10 REM L'ISTRUZIONE "DIM A$(2),B$(2)"
   PUÒ ESSERE ESEGUITA SOLO UNA VOLTA
20 REM PERCIÒ NON COMPARE IN QUESTO MODULO
30 REM SALVA QUESTO MODULO SU DISCO COME "PROG2"
   USANDO L'OPZIONE A
40 COMMON A$( ),B$( )
50 PRINT: PRINT A$(1); A$(2)
```

```
60 B$(1) = "NOTE HOW THE OPTION OF SPECIFYING
   A STARTING LINE NUMBER"
70 B$(2) = "WHEN CHAINING AVOIDS THE DIMENSION
   STATEMENT IN 'PROG1'."
80 CHAIN "PROG1",90
90 END
```

Esempio 2

Nell'esempio che segue vengono illustrate le opzioni MERGE, ALL e DELETE. Dopo il caricamento di A$, nel primo programma, il controllo passa alla linea 1010 del secondo. Alla linea 1040 del secondo programma, il controllo passa alla linea 1010 del terzo programma, mantenendo tutte le variabili e cancellando tutte le linee del secondo programma. Questo processo può essere osservato attraverso il testo descrittivo che viene visualizzato mentre viene eseguito il programma.

```
10 REM QUESTO PROGRAMMA MOSTRA L'USO DI CHAIN CON
   LE OPZIONI MERGE, ALL E DELETE
20 REM SALVA QUESTO MODULO SU DISCO
   COME "MAINPRG"
30 A$ = "MAINPRG"
40 CHAIN MERGE "OVRLAY1",1010,ALL
50 END

1000 REM SALVA QUESTO MODULO SU DISCO COME
     "OVRLAY1" USANDO L'OPZIONE A
1010 PRINT A$ "HAS CHAINED TO OVRLAY1."
1020 A$ = "OVRLAY1"
1030 B$ = "OVRLAY2"
1040 CHAIN MERGE "OVRLAY2",1010, ALL,
     DELETE 1000-1050
1050 END

1000 REM SALVA QUESTO MODULO SU DISCO COME
     "OVRLAY2" USANDO L'OPZIONE A
1010 PRINT A$; "HAS CHAINED TO"; B$;"."
1020 END
```

CHDIR Comando

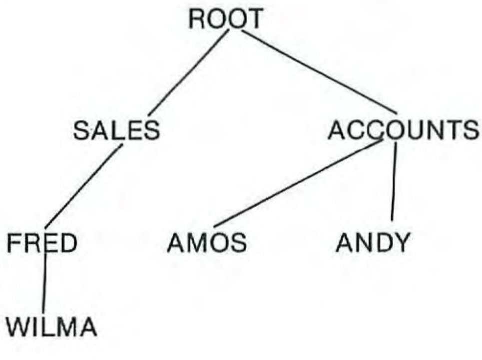
Cambia la directory corrente.

CHDIR *pathname*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
pathname	È un'espressione stringa che identifica la nuova directory che deve diventare la directory corrente.

Esempi



Ammettendo che il diskette sul drive B abbia la struttura di directory illustrata sopra, l'utente può spostare la directory corrente da ROOT a ACCOUNTS:

CHDIR "B:\ACCOUNTS"

ACCOUNTING adesso è la directory corrente sul drive B.

Per spostare la directory corrente da ACCOUNTS a ANDY:

CHDIR "ANDY"

Si evitino troppe nidificazioni di directory come risultato dell'uso ripetuto di MKDIR e CHDIR.

Errori Possibili

"Bad file name": filename sbagliato

"Path not found": pathname non trovato

"Path/File access error": errore d'accesso al pathname/file

CHR$ Funzione

Restituisce una stringa di un carattere il cui codice decimale ASCII costituisce il valore dell'argomento.

CHR$(*n*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È un'espressione intera che deve essere compresa tra 0 e 255 e rappresenta un codice ASCII. Se *n* si trova fuori da tale intervallo, viene visualizzato il messaggio d'errore "Illegal function call".

Caratteristiche

CHR$ viene normalmente usata per inviare un carattere speciale a un device di output. Per esempio, può essere inviato il carattere BEL (CHR$(7)) prima di un messaggio di errore o può essere inviato un salto pagina (CHR$(12)) per cancellare il contenuto del video e riportare il cursore alla posizione iniziale (home) in alto a sinistra. Inoltre è possibile usare PRINT CHR$(*n*) (*n* codice ASCII) per visualizzare un carattere ASCII.

Esempio

```
100 PRINT CHR$(7) 'BEEP
150 PRINT CHR$(LINEFEED%)
200 IF CHR$(INP(IN.PORT%))="A" THEN GOSUB 100
```

CINT Funzione

Converte qualsiasi argomento di tipo numerico in un intero, arrotondandone la parte frazionaria.

CINT(*numexp*)

Caratteristiche

Se *numexp* non è compreso tra i valori -32768 e 32767 si verifica l'errore "Overflow".

Se la parte frazionaria di *numexp* è > = .5 la parte intera viene arrotondata per eccesso altrimenti viene effettuato un troncamento.

Per dettagli sul modo di convertire numeri, rispettivamente in tipi di dati in doppia o semplice precisione, occore fare riferimento alle funzioni CDBL e CSNG e alle funzioni FIX e INT, che forniscono entrambe valori interi.

Esempio

```
Ok
PRINT CINT(45.67)
 46
Ok
PRINT CINT(-3,71)
-4
Ok
```

CIRCLE Istruzione

Traccia una circonferenza (o un'ellisse) avente centro e raggio specificati (solo modo grafica).

CIRCLE [STEP] (*x*,*y*),*radius*[,*color*[,*start*,*end*[,*aspect*]]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
x,y	Sono espressioni numeriche che indicano le coordinate del centro della circonferenza (o ellisse). Esse possono essere coordinate assolute o relative (se viene inclusa l'opzione STEP).
radius	È un'espressione numerica che restituisce un valore intero positivo. Rappresenta il valore del raggio della circonferenza o l'asse maggiore dell'ellisse. *radius* viene misurato orizzontalmente se *aspect*<1 e verticalmente se *aspect*>1. Il valore del raggio è in pixel se non è stata eseguita un'istruzione WINDOW, altrimenti è in coordinate logiche.
color	È un'espressione intera compresa tra 0 e 3 e rappresenta il numero di colore della circonferenza (o ellisse). In Media Risoluzione, *color* sceglie il colore dalla tavolozza selezionata. In Alta e Altissima Risoluzione il valore 0 o 2 di *color* indica il nero mentre 1 o 3 indica il bianco. Se *color* viene omesso, il colore è fornito dal parametro *gforegound* dell'istruzione COLOR; se anche tale parametro è omesso viene assunto il valore 3 in Media Risoluzione e 1 in Alta e Altissima Risoluzione.

ELEMENTO DI SINTASSI	SIGNIFICATO
start,end	Sono espressioni numeriche che specificano angoli in radianti. I valori che possono assumere sono compresi tra -2*PI e 2*PI, dove PI = 3,141593. Esse determinano dove deve iniziare e terminare l'arco della circonferenza (o dell'ellisse).
aspect	È un'espressione numerica il cui valore è un numero reale positivo. A causa della diversa distanza tra un pixel e il successivo lungo i due assi X e Y del video, è necessario specificare un valore di questo parametro per tracciare una circonferenza con monitor diversi. Il valore di default di *aspect* è di 5/6 in Media e Altissima Risoluzione e 5/12 in Alta Risoluzione e dà luogo ad una circonferenza con monitor standard.

Come Tracciare Circonferenze ed Ellissi

L'istruzione CIRCLE traccia circonferenze, se non viene specificato il parametro *aspect*, ed ellissi, se viene specificato un valore di *aspect* diverso dal valore di default (5/6 in Media e Altissima Risoluzione e 5/12 in Alta Risoluzione).

Il parametro *aspect* può essere considerato una frazione atta a specificare il numero di pixel necessari per avere un segmento di uguale misura lungo i due assi. Il numeratore specifica il numero di pixel lungo l'asse Y e il denominatore il numero di pixel lungo l'asse X.

Se *aspect* è minore di uno, *radius* è il raggio rispetto all'asse X, cioé il raggio viene misurato in pixel orizzontalmente e GW-BASIC

traccia delle ellissi con la stessa larghezza e diversa altezza, al variare del parametro *aspect* come riportato nella seguente figura.

Se *aspect* è maggiore di uno, *radius* è il raggio rispetto all'asse Y e GW-BASIC traccia ellissi con la stessa altezza e diversa larghezza, al variare del parametro *aspect*.

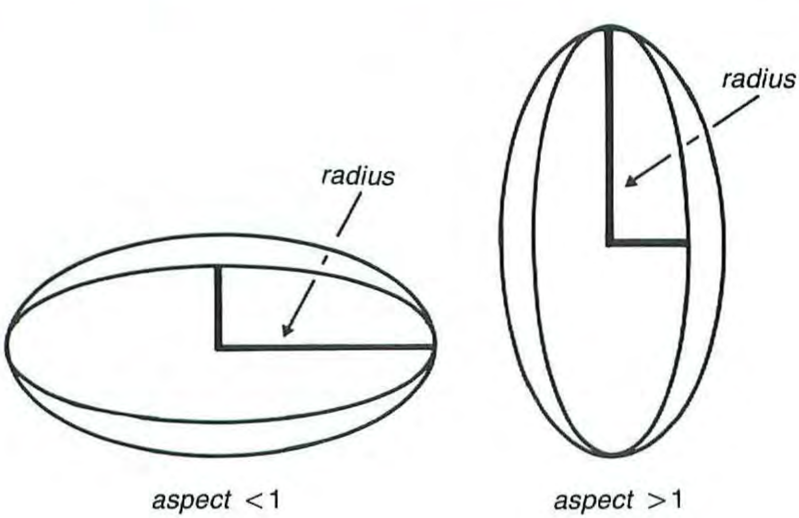


Fig. 8-1 Parametro aspect

Non è possibile trasformare un cerchio (visualizzato con CIRCLE) in un'ellisse usando le istruzioni WINDOW e VIEW (variando il rapporto tra i lati della finestra fisica); è possibile ottenere questo se il cerchio è stato visualizzato (usando l'istruzione LINE) come un poligono regolare con un gran numero di lati.

Per esempio:

```
10 SCREEN 1
...
100 CIRCLE (100,150),50,,,,5/18
```

traccia un'ellisse orizzontale con un raggio rispetto all'asse X di 50 pixel.

Come Tracciare Archi

L'istruzione CIRCLE può anche tracciare solo parte di una circonferenza (o di un'ellisse) e cioé un arco.

Per tracciare un arco è necessario introdurre i parametri *start* e *end*. Essi specificano, in radianti, i due estremi dell'arco.

Gli angoli vengono misurati a partire dalla parallela all'asse X, passante per il cerchio, e procedendo in senso antiorario, come riportato nella Figura 8-2.

Per esempio, l'istruzione seguente specifica solo un quarto di circonferenza:

```
10 CIRCLE (100,150),50,1,0,3.141593/2
```

Gli angoli devono essere misurati in radianti. Se gli angoli sono misurati in gradi devono essere convertiti in radianti prima dell'esecuzione dell'istruzione CIRCLE. Per convertire la misura degli angoli da gradi in radianti si deve moltiplicare per 0.0174532.

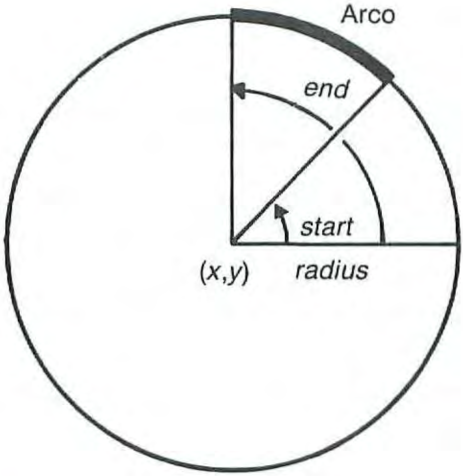


Fig. 8-2 Arco di Cerchio

Come Tracciare Raggi

L'istruzione CIRCLE può tracciare un raggio dal centro verso uno dei due estremi dell'arco.

Un estremo negativo genera un raggio verso quell'estremo. L'estremo -0 non viene considerato un estremo negativo. Per superare questa limitazione è possibile usare un numero negativo piccolo (per esempio -0.001 invece di -0). Quando entrambi gli estremi sono negativi, vengono tracciati entrambi i raggi. Il segno meno non influenza l'arco in sé, cioé gli angoli vengono considerati positivi. Si noti che questo procedimento è diverso da quello che consiste nell'aggiungere 2*PI (dove PI è 3.141593). L'angolo di partenza può essere maggiore o minore dell'angolo finale. Per esempio:

```
10 SCREEN 1
...

100 CIRCLE (100,150),50,1,-0.001,-3.141593/2
```

traccia un quarto di circonferenza delimitato da due raggi.

Ultimo Punto Riferito

L'ultimo punto riferito dopo che una circonferenza (o un'ellisse) è stata tracciata è il centro della circonferenza (o dell'ellisse).

Opzione STEP

Le coordinate possono essere assolute o relative (se viene specificata l'opzione STEP). Le coordinate relative rappresentano l'incremento rispetto all'ultimo punto riferito.

Per esempio, se l'ultimo punto riferito è 100,50 allora:

```
50 CIRCLE (200,200),50
```

```
50 CIRCLE STEP (100,150),50
```

tracciano entrambe una circonferenza con centro (200,200) e raggio 50. Il primo esempio usa coordinate assolute; il secondo coordinate relative.

Esempio

Nell'esempio che segue vengono tracciate tre circonferenze che si intersecano e viene colorata l'area di intersezione.

```
 5 SCREEN 1
10 COLOR 0,0,3,0
20 CLS
30 CIRCLE (100,120),90
40 CIRCLE (150,130),120
50 CIRCLE (250,120),100
60 PAINT (180,120)
```

CLEAR Comando

Azzera tutte le variabili numeriche, inizializza tutte le variabili stringa al valore stringa nulla, chiude tutti i file aperti e annulla l'effetto di ogni istruzione DIM o DEF. Vi sono opzioni che permettono di definire la memoria massima disponibile per GW-BASIC e lo spazio dedicato allo stack.

CLEAR [,[*memory*][,*stack*]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
memory	È un'espressione intera che rappresenta la più alta locazione di memoria disponibile per GW-BASIC (cioé l'estensione massima del Segmento Dati GW-BASIC). Il valore massimo è 65536. Il valore di default (quando il parametro è omesso o vale 0) è il valore massimo corrente della memoria come specificato dal comando GWBASIC oppure è 65536 se tale comando non specifica un valore.
stack	È un'espressione intera il cui valore rappresenta lo spazio riservato allo stack. Il valore di default è il valore minore tra 128 byte e un ottavo della memoria disponibile.

Caratteristiche

Il parametro *memory* viene utilizzato per riservare sufficiente spazio in memoria per il programma o per limitare lo spazio riservato al programma al fine di allocare routine in linguaggio macchina nell'area di lavoro GW-BASIC. Il parametro *stack* viene utilizzato per ampliare lo spazio riservato allo stack; questo spazio può essere critico quando un programma ha molte istruzioni GOSUB nidificate e/o molti cicli FOR...NEXT nidificati e/o istruzioni PAINT per colorare figure complesse.

Si può verificare l'errore "Out of memory" se non c'è spazio sufficiente per il programma o per lo stack.

GW-BASIC alloca in modo dinamico lo spazio riservato alle stringhe. Viene visualizzato il messaggio di errore "Out of string space" solo se non vi è più uno spazio in memoria disponibile per GW-BASIC.

Il comando CLEAR lascia inalterato il programma in memoria e svolge le seguenti operazioni:

- Chiude tutti i file.
- Azzera tutte le variabili COMMON.
- Azzera lo stack e libera lo spazio riservato alle stringhe.
- Azzera tutte le variabili numeriche semplici e gli elementi di vettore di tipo numerico.
- Inizializza al valore stringa nulla tutte le variabili stringa semplici e gli elementi di vettore di tipo stringa.
- Cancella tutti i buffer su disco.
- Annulla l'effetto di ogni istruzione DIM o DEF (DEF FN, DEFINIT/SNG/DBL/STR, DEF SEG e DEF USR) eseguita in precedenza.

Dimensiona l'area di memoria riservata a GW-BASIC e allo stack (se vengono specificati i parametri *memory* e *stack*).

Esempi

```
CLEAR

CLEAR ,32768

CLEAR ,,2000

CLEAR ,32768,2000
```

CLOSE Istruzione

Chiude le operazioni di I/O su un file o un device. L'istruzione CLOSE viene generalmente usata all'interno di un programma.

CLOSE [[#]*filenum*[,[#]*filenum*]...]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filenum	È il numero con cui il file è stato aperto. Un'istruzione CLOSE senza argomenti chiude tutti i file aperti.

Caratteristiche

Un file rimane associato ad un numero fino all'esecuzione di un'istruzione CLOSE. Il file può essere riaperto tramite un'istruzione OPEN con lo stesso numero o con uno diverso; allo stesso modo un numero di file può essere riutilizzato per riaprire un altro file.

Un'istruzione CLOSE per un file di output sequenziale scrive il buffer di output finale.

L'istruzione END e il comando NEW chiudono sempre automaticamente tutti i file su disco, l'istruzione STOP non chiude i file su disco.

Esempio

Per leggere i dati di un file sequenziale aperto per operazioni di output o di append è necessario prima chiudere il file (istruzione CLOSE) e poi riaprirlo (istruzione OPEN) nel modo "I".

```
100 CLOSE #1
110 OPEN "I",#1,"DATA"
```

CLS Istruzione

Cancella tutto il contenuto del video o di una finestra.

CLS [*n*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È un'espressione intera il cui valore è compreso nell'intervallo da 0 a 2.

Caratteristiche

L'istruzione CLS senza parametri cancella il contenuto dell'intero video e colora quest'ultimo con il colore di fondo per il testo, se non è stata definita una finestra fisica, e reinserisce la linea dei tasti funzione (se è stata abilitata la visualizzazione dei tasti funzione). Se è stata definita una finestra fisica, solo quella corrente viene cancellata e colorata con il colore di fondo per la grafica. Con l'invio di un carattere di salto pagina (digitando **CTRL L PRINT CHR$(12)**) si ottiene lo stesso risultato. Se è presente una finestra di testo e non è presente nessuna finestra fisica (quindi non è stata usata nessuna istruzione VIEW) l'istruzione CLS cancella solamente la finestra di testo.

CLS 0 cancella l'intero contenuto del video, ripristinando la visualizzazione dei tasti funzione.

CLS 1 cancella il contenuto della finestra fisica corrente e la colora con il colore di fondo per la grafica. Se non è stata definita nessuna finestra fisica questa istruzione non provoca nessun effetto.

CLS 2 cancella il contenuto di una finestra di testo e la colora con il colore di fondo per il testo, senza ripristinare la visualizzazione di tasti funzione.

Osservazioni

L'istruzione CLS non solo cancella tutto il video o una sua parte, ma riporta anche il cursore nell'angolo del video in alto a sinistra (nel Modo Testo).

Nel modo grafica l'istruzione CLS fa sì che l'ultimo punto riferito sia il punto centrale del video. Il contenuto del video può anche essere cancellato premendo **CTRL HOME** o modificando il modo video, tramite l'istruzione SCREEN, o l'ampiezza video, tramite l'istruzione WIDTH.

Esempi

```
10  CLS
```

Cancella il contenuto del video (o della finestra corrente o della finestra di testo)

```
60  CLS 0
```

Cancella il contenuto dell'intero video

```
90  CLS 1
```

Cancella il contenuto della finestra fisica e la colora con il colore di fondo per la grafica

```
110  CLS 2
```

Cancella il contenuto della finestra di testo e la colora con il colore di fondo per il testo.

COLOR Istruzione (Modo Testo)

Imposta il colore di primo piano e il colore di fondo per il testo (solo Modo Testo).

COLOR [*foreground*][,[*background*][,*dummy*]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
foreground	È un'espressione numerica arrotondata all'intero piú vicino. Deve essere compresa tra 0 e 31. Valori superiori a 15 sono interpretati modulo 16. Seleziona il colore di primo piano dei caratteri.
background	È un'espressione numerica arrotondata all'intero più vicino. Deve essere compresa tra 0 e 15, ma è interpretata modulo 8 e quindi il valore massimo è 7. Seleziona il colore di fondo dei caratteri.
dummy	Questo parametro è presente per compatibilità con altri linguaggi BASIC, ma non ha alcun effetto. Può specificare il colore del margine del video su altri sistemi.

Modo Testo a Colori

Se è presente il video a colori, è possibile usare i seguenti colori di primo piano dei caratteri:

0 Nero
1 Blu
2 Verde
3 Ciano
4 Rosso
5 Magenta
6 Giallo
7 Bianco
8 Grigio
9 Blu chiaro
10 Verde chiaro
11 Ciano chiaro
12 Rosso chiaro
13 Magenta charo
14 Giallo chiaro
15 Bianco ad alta intensità.

È possibile far lampeggiare i caratteri aggiungendo 16 al codice del colore di primo piano (*foreground*) desiderato.

Per il colore di fondo (*background*) si possono selezionare solo i colori da 0 a 7.

Modo Testo in B/N

In un sistema monocromatico si possono utilizzare i seguenti valori per il colore di primo piano (*foreground*):

0 Nero

1 Caratteri sottolineati con colore di primo piano bianco

2-6 Tonalità del grigio

7 Bianco

L'aggiunta di 8 al codice di colore dà la possibilità di ottenere quel colore ad alta intensità. Per esempio, il 15 equivale al bianco ad alta intensità.

È possibile far lampeggiare il carattere aggiungendo 16 al codice del colore di primo piano desiderato.

Per il colore di fondo (*background*) si possono selezionare i seguenti valori:

0-1 Nero

2-6 Tonalità del grigio

7 Bianco

Osservazioni

I colori di primo piano e di fondo dei caratteri possono essere uguali. In questo caso qualsiasi carattere visualizzato risulta invisibile.

Può essere omesso qualsiasi parametro. I parametri omessi assumono il valore precedente.

Dopo l'inizializzazione i valori di default sono i seguenti:

- *foreground* = 7 (Bianco)
- *background* = 0 (Nero)

Significa che, se nel programma non esiste alcuna istruzione COLOR, il sistema assume:

```
COLOR 7,0
```

Esempi

```
100 COLOR 0,2
```

Seleziona un colore di primo piano nero e un colore di fondo verde con video a colori; un colore di primo piano nero e un colore di fondo grigio con video B/N.

```
150 COLOR 15,1
```

Seleziona un colore di primo piano bianco ad alta intensità e un colore di fondo blu con video a colori; un colore di primo piano bianco ad alta intensità su un colore di fondo nero con video B/N.

Errori Possibili

Se l'istruzione COLOR termina con una virgola (,) viene generato l'errore "Missing operand". Per esempio:

```
COLOR 2,
```

non è valido.

Se ai parametri viene assegnato un valore fuori dall'intervallo specificato si genera l'errore "Illegal function call". In questo caso vengono conservati i valori precedenti.

COLOR Istruzione (Grafica a Media Risoluzione)

Consente di scegliere la tavolozza e di specificare il colore di primo piano di default e il colore di fondo per la grafica. Permette inoltre di specificare il colore di primo piano e di fondo per il testo.

COLOR [*background*][,[*palette*][,[*gforeground*]
[,[*gbackground*][,*tforeground*]]]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
background	È un'espressione numerica arrotondata all'intero più vicino. Deve essere compresa tra 0 e 31. Valori superiori a15 sono interpretati modulo 16. Rappresenta il codice del colore di fondo dei caratteri. È anche il colore corripondente al numero di colore 0 che può essere specificato in una istruzione di grafica. Questo parametro può anche specificare l'intensità del colore di primo piano in grafica (intensità dei pixel con numero di colore 1, 2 o 3). Infatti se il parametro è maggiore di 15 viene selezionato il colore di primo piano ad alta intensità. Se *background* non viene specificato il suo valore di default è 0.

ELEMENTO DI SINTASSI	SIGNIFICATO
palette	È un'espressione numerica arrotondata all'intero piú vicino. Deve essere compresa tra 0 e 255. Essa seleziona una delle due possibili tavolozze di colori. Una istruzione di grafica può specificare il colore di primo piano tramite un numero di colore (0, 1, 2 o 3) che seleziona il colore desiderato in funzione della tavolozza attiva. Se il numero di colore è 0 viene selezionato il colore specificato da *background*. Viene selezionata la tavolozza 0 quando *palette* è un numero pari, mentre viene selezionata la tavolozza 1 se *palette* è un numero dispari.
gforeground	È un'espressione numerica arrotondata all'intero più vicino e deve essere compresa tra 0 e 7. Valori superiori a 3 sono interpretati modulo 4. Seleziona il colore di primo piano di default per la grafica, assunto quando non viene specificato da nessuna istruzione di grafica. Se viene omesso *gforeground*, si assume il numero di colore 3. Se l'istruzione SCREEN seleziona un nuovo modo video 1, *gforeground* viene reinizializzato al valore di default (3). Un valore superiore a 3 causa un'operazione XOR tra i colori dei pixel su video e i colori dei punti da visualizzare.
gbackground	È un'espressione numerica arrotondata all'intero più vicino e deve essere compresa tra 0 e 3. Seleziona il colore di

ELEMENTO DI SINTASSI	SIGNIFICATO
	fondo per la grafica usato quando si cancella il contenuto di una finestra fisica con CLS 1, o semplicemente con CLS. Il valore di default è 0 e viene sempre adottato quando si seleziona un nuovo modo video. L'istruzione CLS 1 cancella una finestra solo se è stata esplicitamente definita con una istruzione VIEW.
tforeground	È un'espressione numerica arrotondata all'intero più vicino e deve essere compresa tra 0 e 7. Valori superiori a 3 sono interpretati modulo 4. Seleziona il colore di primo piano dei caratteri. Assume il valore di default 3. Un valore superiore a 3 provoca l'operazione XOR tra i colori dei pixel su video e il colore dei caratteri da visualizzare.

Osservazioni

Quando viene impostata un'istruzione CIRCLE, DRAW, LINE, PAINT, PRESET o PSET è possibile specificare un numero di colore (0, 1, 2, o 3). Questo parametro seleziona il colore dalla tavolozza corrente come definito dall'istruzione COLOR. Esistono due tavolozze, ciascuna di quattro colori, come descritto nel Capitolo 5.

Se il numero di colore non è stato specificato viene assunto il colore di primo piano di default per la grafica (cioé, il valore di *gforeground*, o 3 se *gforeground* non è stato specificato).

Quando viene visualizzato un testo, il colore di primo piano dei caratteri viene definito da *tforeground* (il cui valore di default è 3) ed il colore di fondo dei caratteri viene definito da *background* (il cui valore di default è 0, cioé nero).

Nell'istruzione COLOR può essere omesso qualsiasi parametro. I parametri omessi assumono i valori precedenti.

Dopo l'inizializzazione i valori di default sono:

- *background* = 0 (nero)
- *palette* = 1 (ciano, magenta, bianco)
- *gforeground* = 3 (bianco)
- *gbackground* = 0 (nero)
- *tforeground* = 3 (bianco)

Significa che, se nel programma non esiste alcuna istruzione COLOR, il sistema assume:

```
COLOR 0,1,3,0,3
```

Esempi

```
10 SCREEN 1,0
20 COLOR 10,1,2,0
```

Sceglie come colore di fondo dei caratteri il verde chiaro e seleziona la tavolozza 1 (ciano, magenta, bianco), imposta il colore di primo piano di default (magenta) e il colore di fondo (verde chiaro) per la grafica.

```
100 COLOR 0
```

Il colore di fondo dei caratteri rimane verde chiaro e viene selezionata la tavolozza 0 (verde, rosso, giallo).

COLOR Istruzione (Grafica ad Alta Risoluzione)

Definisce il colore di primo piano di default e il colore di fondo per la grafica. Può inoltre specificare se i caratteri devono essere visualizzati in negativo, in positivo o tramite una operazione XOR con i pixel su video.

COLOR [*gforeground*][,[*gbackground*][,*tforeground*]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
gforeground	È un'espressione numerica arrotondata all'intero più vicino. Deve essere compresa tra 0 e 3. Rappresenta un numero di colore e seleziona il colore di primo piano di default per la grafica; cioé il colore di primo piano assunto quando un'istruzione di grafica non specifica alcun colore. Se *gforeground* è omesso, viene assunto 1 (bianco). I valori 0 (nero) oppure 1 (bianco) rappresentano colori effettivi; i valori 2 e 3 specificano che deve essere eseguita un'operazione XOR tra i colori dei pixel su video e i colori dei pixel da visualizzare.
gbackground	È un'espressione numerica, arrotondata all'intero piú vicino, il cui valore può

ELEMENTO DI SINTASSI	SIGNIFICATO
	essere 0 (nero) o 1 (bianco). Specifica il colore di fondo per la grafica usato per cancellare il contenuto delle finestre fisiche. Il valore di default è 0.
tforeground	È un'espressione numerica, arrotondata all'intero più vicino. Un risultato uguale a 0 indica caratteri in negativo (scritti con pixel neri su pixel di fondo bianchi). Un risultato uguale a 1 indica caratteri in positivo (scritti con pixel bianchi su pixel di fondo neri); questo è il valore di default. Un risultato maggiore di 1 indica che deve essere eseguita un'operazione XOR tra i colori dei pixel presenti su video e i colori dei pixel dei caratteri da visualizzare.

Caratteristiche

Quando si inserisce un'istruzione CIRCLE, DRAW, LINE, PAINT, PRESET o PSET in un programma è possibile specificare un numero di colore pari a 0, 1, 2 o 3. In Alta Risoluzione il numero di colore 0 indica il nero e 1 indica il bianco. Il numero di colore 2 viene trattato come 0 ed il numero di colore 3 come 1.

Se non si specifica un numero di colore, il colore di primo piano di default per la grafica viene selezionato da *gforeground* oppure è 1 (se *gforeground* non è stato specificato). Quando si visualizza un testo, se non viene specificato *tforeground* uguale a 0 (caratteri in negativo), il colore di primo piano è 1 (bianco) e il colore di fondo è 0 (nero).

È anche possibile specificare un'operazione XOR tra i pixel su video ed i pixel in memoria, se il valore di *tforeground* è maggiore di 1.

Nell'istruzione COLOR è possibile omettere qualsiasi parametro. I parametri omessi assumono i valori precedenti.

Dopo l'inizializzazione, i valori di default sono i seguenti:

- *gforeground* = 1 (bianco)
- *gbackground* = 0 (nero)
- *tforeground* = 0 (nessuna operazione XOR per il testo)

Significa che, se non esiste alcuna istruzione COLOR nel programma, il sistema assume:

COLOR 1,0,0

Esempio

SCREEN 2
COLOR 0,1,0

Viene selezionato il nero come colore di primo piano di default per la grafica, un colore di fondo bianco e nessuna operazione XOR.

COLOR Istruzione (Grafica ad Altissima Risoluzione)

Definisce il colore di primo piano di default e il colore di fondo per la grafica. Specifica inoltre se visualizzare i caratteri in negativo, in positivo o con una operazione XOR con i pixel su video.

COLOR [*gforeground*][,[*gbackground*][,*tforeground*]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
gforeground	Come per il modo ad Alta Risoluzione.
gbackground	Come per il modo ad Alta Risoluzione.
tforeground	È un'espressione numerica arrotondata all'intero più vicino. Un risultato uguale a 0 indica caratteri in negativo (scritti con pixel neri su pixel di fondo bianchi). Un risultato uguale a 1 indica caratteri in positivo (scritti con pixel bianchi su pixel di fondo neri); questo è il valore di default. Un risultato maggiore di 1 indica che viene fatta un'operazione XOR tra i colori dei pixel presenti su video e i colori dei pixel dei caratteri da visualizzare.

Caratteristiche

Quando viene inserita un'istruzione CIRCLE, DRAW, LINE, PAINT, PRESET O PSET in un programma è possibile specificare un numero di colore pari a 0, 1, 2 o 3. Il numero di colore 0 indica il nero e 1 il bianco. Il numero di colore 2 viene trattato come 0 e 3 come 1.

Se non viene specificato un numero di colore, il colore di primo piano di default per la grafica viene selezionato da *gforeground* oppure è 1 se tale parametro non è stato specificato.

Quando si visualizza un testo, se non viene specificato *tforeground* uguale a 0 (caratteri in negativo), il colore di primo piano dei caratteri è 1 (bianco) e il colore di fondo è 0 (nero).

Tramite l'istruzione COLOR è anche possibile specificare un'operazione XOR tra i colori dei pixel su video ed i colori dei pixel in memoria.

Nell'istruzione COLOR è possibile omettere qualsiasi parametro. I parametri omessi assumono i valori precedenti. Dopo l'inizializzazione i valori di default sono i seguenti:

- *background* = 1 (bianco)
- *gbackground* = 0 (nero)
- *tforeground* = 1 (video normale, nessuna operazione XOR)

Significa che, se nel programma non esiste alcuna istruzione COLOR, il sistema assume:

```
COLOR 1,0,1
```

Esempio

```
SCREEN 3
COLOR 0,1,0
```

Viene selezionato il nero come colore di primo piano per la grafica su un colore di fondo bianco, i caratteri in negativo e nessuna operazione XOR.

COM(*n*) Istruzione

COM(*n*) ON abilita, COM(*n*) OFF disabilita e COM(*n*) STOP sospende il rilevamento di eventi riguardanti l'attività di comunicazione sul canale specificato.

COM(*n*) { ON | OFF | STOP }

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È un'espressione intera che specifica il numero del canale di comunicazione, che può essere 1 o 2.

Per Abilitare o Disabilitare il Rilevamento di Eventi di Comunicazione

SE...	ALLORA...
COM(*n*) ON viene eseguita	viene abilitato il rilevamento di eventi di comunicazione. Quando questo è abilitato, se nell'istruzione ON COM(*n*) GOSUB viene speficifato un numero di linea diverso da zero, GW-BASIC controlla, ogni volta che viene eseguita ▶

SE...	ALLORA...
	una nuova istruzione, se qualche carattere è presente nel canale di comunicazione. In questo caso viene eseguita l'istruzione ON COM(*n*) GOSUB.
COM(*n*) OFF viene eseguita	viene disabilitato il rilevamento di eventi di comunicazione. Nessun evento viene quindi memorizzato.
COM(*n*) STOP viene eseguita	viene disabilitato il rilevamento di eventi di comunicazione, ma se viene svolta un'attività di comunicazione, essa viene memorizzata e, quando il rilevamento di eventi verrà abilitato, verrà eseguita l'istruzione ON COM(*n*) GOSUB.
ON COM(*n*) GOSUB viene eseguita	viene sospeso il rilevamento di eventi di comunicazione.
ha luogo un rilevamento di errori	tutte le routine di rilevamento vengono automaticamente disabilitate (compreso il rilevamento di ERROR).

Esempio

```
10 COM(1) ON
```

Abilita il rilevamento di errori riguardanti l'attività di comunicazione sul canale 1.

COMMON Istruzione

Definisce un'area comune a più programmi che non viene cancellata dal programma concatenato e permette il trasferimento di variabili da un programma all'altro.
L'istruzione COMMON può essere usata solamente all'interno di un programma.

COMMON *variable*[,*variable*]...

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
variable	È il nome di una variabile numerica o stringa, che deve essere passata al programma concatenato. Le variabili indicanti una matrice vengono specificate aggiungendo "( )" al nome di variabile.

Caratteristiche

L'istruzione COMMON viene usata unitamente all'istruzione CHAIN. Le istruzioni COMMON possono essere usate in qualsiasi punto del programma, ma è consigliabile usarle all'inizio.

Le variabili specificate nell'istruzione COMMON vengono allocate nell'area comune, a partire dall'inizio dell'area e seguendo l'ordine di scrittura con cui compaiono nel programma.

Il programma concatenato non deve necessariamente specificare, tramite istruzione COMMON, le variabili comuni specificate dal programma chiamante; ma farà riferimento a queste variabili usando gli stessi nomi indicati nel programma chiamante. Ogni istruzione di definizione di tipo (DEFINT, DEFSNG, DRFDBL, DEFSTR), relativa a variabili comuni, deve precedere le istruzioni COMMON associate e deve essere ripetuta nel programma concatenato.

Le variabili comuni devono essere sempre inizializzate all'interno del programma chiamante. Le matrici comuni devono essere definite esplicitamente tramite istruzioni DIM, nel programma chiamante (ma non nel programma concatenato, altrimenti si verifica l'errore "Duplicated definition"). Le istruzioni DIM devono precedere le istruzioni COMMON associate.

Esempio 1

```
10 REM PG1
20 COMMON A1,B1,C1,D1$
 .
 .
 .
80 CHAIN "A:PG2"
90 END

10 REM PG2
20 PRINT A1,B1,C1,D1$
 .
 .
 .
120 END
```

Questo esempio mostra che il programma concatenato non deve necessariamente specificare, tramite istruzione COMMON, le variabili comuni, specificate invece dal programma chiamante.

In questo esempio i valori delle variabili A1,B1,C1 e D1$ del programma PG1 vengono passati al programma concatenato PG2, che li visualizza (vedere l'istruzione 20).

Esempio 2

```
10 REM PG1
20 DEFDBL C
30 COMMON A1,B1,C1,D1$
 .
 .
 .
90 CHAIN "A:PG2"
100 END

10 REM PG2
20 DEFDBL C
 .
 .
 .
130 END
```

Ogni istruzione di definizione di tipo (DEFINIT, DEFSNG, DEFDBL, DEFSTR) relativa a variabili comuni deve precedere l'istruzione COMMON associata e deve essere ripetuta nel programma concatenato. (Notare la presenza delle istruzioni DEFDBL nei due programmi PG1 e PG2.)

Esempio 3

```
10 REM PROGRAM1
20 COMMON A$,B$,C$
30 COMMON A$,A1
 .
 .
 .
100 END
```

È buona norma di programmazione non ripetere lo stesso nome di variabile (in questo caso A$) in istruzioni COMMON diverse

nell'ambito dello stesso programma o nella stessa istruzione COMMON. In ogni caso le definizioni multiple sono equivalenti a una singola.

Esempio 4

```
10 REM PG1
20 DIM A1(15,20)
30 COMMON A1( ),B1,C1
 .
 .
 .
100 CHAIN "A:PG2"
110 END

10 REM PG2
 .
 .
 .
50 PRINT A1(1,1)
 .
 .
 .
90 END
```

Questo esempio mostra la definizione e il trasferimento di una matrice, tramite l'istruzione COMMON, al programma concatenato.

Esempio 5

```
10 REM mod1
20 A=1:B=2
30 COMMON A,B
40 GOTO 60
50 COMMON C
60 CHAIN "mod3"
```

```
10 REM mod2
20 A=1:B=2
30 COMMON A
40 GOTO 60
50 COMMON B
60 CHAIN "mod3"
```

```
10 REM mod3
20 PRINT A;B
```

L'istruzione COMMON è di tipo dichiarativo ed alloca un'area comune anche nel caso in cui il controllo dell'esecuzione non venga mai passato ad essa.

Per esempio, quando si esegue "mod1" si verifica l'errore "Illegal function call in 50", perché la variabile C non è stata inizializzata. Quando, invece, si esegue "mod2", viene concatenato il programma "mod3", che visualizza sia la variabile A che la variabile B, anche se l'istruzione 50 di "mod2" è stata saltata.

CONT Comando

Continua l'esecuzione di un programma dopo la digitazione di **CTRL BREAK** o l'esecuzione di un'istruzione STOP o END. Il comando CONT deve essere usato solo in modo immediato.

CONT

Caratteristiche

L'esecuzione riprende dal punto in cui si è verificata l'interruzione. Se quest'ultima si è verificata dopo un prompt di un'istruzione INPUT, l'esecuzione continua ristampando il prompt ("?" o la stringa di prompt).

Il comando CONT viene generalmente usato unito all'interruzione STOP per operazioni di debugging. Quando l'esecuzione viene interrotta i valori intermedi possono essere esaminati e modificati con l'uso delle istruzioni immediate. L'esecuzione può essere ripresa impostando il comando CONT o l'istruzione immediata GOTO, che rimanda il controllo dell'esecuzione ad un numero di linea specificato.

Il comando CONT non può essere usato per continuare l'esecuzione dopo che si è verificato un errore o se il programma è stato modificato durante l'interruzione.

Esempio

```
10 INPUT A,B
20 TEMP=A*B
30 STOP
40 FINAL = TEMP+300: PRINT FINAL
RUN
? 32 2.4
Break in 30
Ok
PRINT TEMP
 76.8
Ok
CONT
 376.8
Ok
```

COS Funzione

Calcola il coseno dell'argomento.

COS(*numexp*)

Caratteristiche

L'argomento *numexp* viene assunto come la misura dell'angolo espressa in radianti.

Se non viene specificata l'opzione /D nel comando GWBASIC il calcolo della funzione COS viene effettuato in semplice precisione.

Esempio

```
10 X=2*COS(.4)
20 PRINT X
RUN
 1.842122
Ok
```

CSNG Funzione

Converte un argomento di tipo numerico in un numero in semplice precisione.

CSNG(*numexp*)

Osservazioni

Si faccia riferimento alle funzioni CINT e CDBL per la conversione di numeri nei formati intero e doppia precisione, rispettivamente.

Esempio

```
10 A# = 975.3421
20 PRINT A#; CSNG(A#)
RUN
 975.3421020507813  975.3421
Ok
```

CSRLIN Funzione

Restituisce la posizione verticale del cursore.

CSRLIN

Caratteristiche

Il valore restituito dalla funzione CSRLIN è un numero intero compreso tra 1 e 25. Per ottenere la posizione orizzontale del cursore (numero di colonna) si deve usare la funzione POS.

Esempio

```
10 Y = CSRLIN 'Memorizza la riga corrente
20 X = POS (0) 'Memorizza la colonna corrente
30 LOCATE 24,1 :PRINT "HELLO" 'Stampa HELLO sull'ultima
   riga
40 LOCATE Y,X 'Ripristina la posizione di riga, e colonna
   precedenti
```

CVI, CVS, CVD Funzioni

Convertono i valori stringa in valori numerici.

Sintassi 1

CVI(*2-byte string***)**

Sintassi 2

CVS(*4-byte string***)**

Sintassi 3

CVD(*8-byte string***)**

Caratteristiche

I valori che vengono letti dal buffer di un file ad accesso diretto devono essere convertiti da stringa in numerici.

CVI converte una stringa di 2-byte in un numero intero.

CVS converte una stringa di 4-byte in un numero in semplice precisione.

CVD converte una stringa di 8-byte in un numero in doppia precisione

Si faccia riferimento anche alle funzioni MKI$, MKS$, MKD$.

Esempio

```
.
.
.
70 FIELD #1,4 AS N$, 12 AS B$,...
80 GET #1
90 Y = CVS(N$)
.
.
.
```

DATA Istruzione

Crea un file dati interno, cioé una sequenza di dati che devono essere trasferiti nelle variabili di programma tramite un'istruzione READ. Un'istruzione DATA può essere usata solamente all'interno di un programma.

DATA *constant*[,*constant*]...

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
constant	Può essere una costante numerica o stringa. Le costanti numeriche possono essere in qualsiasi formato: intero, esadecimale, ottale, semplice o doppia precisione. Nelle istruzioni DATA le costanti stringa devono essere incluse tra doppi apici solo se contengono virgole, due punti (:) o spazi iniziali o finali significativi; altrimenti tali caratteri non sono necessari.

Caratteristiche

Le istruzioni DATA non sono esecutive e possono essere utilizzate in qualsiasi punto del programma. Un'istruzione DATA può contenere tante costanti, separate da virgole, quante possono essere incluse in una linea. Non ci sono limiti al numero di istruzioni DATA che possono venire usate in un programma.

In un programma un'istruzione DATA non deve necessariamente corrispondere ad un'istruzione READ specifica. Questo perché prima dell'esecuzione di un programma viene creato un file dati che contiene tutti i valori di tutte le istruzioni DATA del programma, secondo le sequenze dei numeri di linea. Nel corso dell'esecuzione del programma, l'istruzione READ acquisisce i suoi valori da questo file.

Il tipo di un dato in ingresso nella sequenza dei dati deve essere coerente con il tipo di variabile cui deve essere assegnato; cioé le variabili numeriche richiedono, come dati, costanti numeriche (è consentita la conversione da un tipo numerico ad un altro, per esempio è possibile avere una costante in semplice precisione

a punto mobile associata a una variabile intera), mentre le variabili stringa richiedono dati stringa tra doppi apici o no.

Una stringa deve essere inclusa tra doppi apici se contiene virgole (ad esempio "BIRMINGHAM, ") oppure spazi iniziali o finali (ad esempio " BIRMINGHAM").

Le istruzioni DATA possono essere rilette dall'inizio tramite l'istruzione RESTORE.

Esempio

```
Ok
10 PRINT "CITY", "STATE", "ZIP"
20 READ C$,S$,Z
30 DATA "BIRMINGHAM,",ALABAMA,12345
40 PRINT C$,S$,Z
RUN
CITY            STATE      ZIP
BIRMINGHAM,     ALABAMA    12345
Ok
```

DATE$ Funzione e Istruzione

Permette di leggere la data (come funzione) oppure di impostare la data (come istruzione).

Sintassi 1: funzione

stringvar = **DATE$**

Sintassi 2: istruzione

DATE$ = *stringexp*

Caratteristiche

DATE$, usata come funzione, acquisisce la data corrente e la assegna alla variabile stringa *stringvar*. La funzione DATE$ può anche essere usata in qualsiasi espressione stringa compresa in un'istruzione LET o PRINT.

DATE$, usata come istruzione, imposta la data corrente. In questo caso DATE$ è l'oggetto di un'assegnazione stringa.

La data può anche essere stata impostata da MS-DOS prima di entrare in ambiente GW-BASIC.

Osservazioni

Se *stringexp* non è una stringa valida si verifica l'errore "Type mismatch". I valori precedenti vengono conservati.

Usata come funzione, DATE$ restituisce una stringa di 10 caratteri nella forma "*mm-dd-yyyy*" dove *mm* rappresenta il mese (da 01 a 12), *dd* rappresenta il giorno (da 01 a 31) e *yyyy* rappresenta l'anno (dal 1980 a 2099).

Se DATE$ viene usata come istruzione, *stringexp* può essere rappresentata in una delle seguenti forme:

"*mm-dd-yy*"
oppure
"*mm/dd/yy*"
oppure
"*mm-dd-yyyy*"
oppure
"*mm/dd/yyyy*"

Se viene digitata una sola cifra per il mese o il giorno, questa viene automaticamente preceduta da uno zero. Se viene digitata una sola cifra (*y*) per l'anno, GW-BASIC interpreta l'anno come 200*y*, se invece vengono digitate due cifre, viene considerato l'anno 19*yy*.

Se un qualsiasi valore si trova fuori dall'intervallo permesso o non esiste, viene emesso il messaggio "Illegal function call". Eventuali date precedenti vengono conservate.

Esempio

```
DATE$ = "07-01-83"
Ok
PRINT DATE$
07-01-1983
Ok
```

La data corrente viene fissata in 1 Luglio 1983.

DEF FN Istruzione

Definisce una funzione utente e le assegna un nome. Un'istruzione DEF FN può essere usata solo all'interno di un programma.

DEF FN*name*[(*argument*,*argument*...)] = *expression*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
name	È un nome di variabile che inizia con FN. Non è possibile inserire spazi tra FN

ELEMENTO DI SINTASSI	SIGNIFICATO
	e la parte rimanente del nome, inoltre il primo carattere dopo FN deve essere una lettera.
argument	È un nome di variabile "fittizio" che viene sostituito da un valore quando la funzione viene richiamata.
expression	Espressione che svolge il calcolo della funzione. Il tipo di espressione deve essere coerente con il tipo di funzione (numerico o stringa) specificato dal parametro *name*.

Caratteristiche

Nell'istruzione DEF FN, i nomi di variabili servono solo a definire la funzione e non interessano le variabili di programma aventi lo stesso nome. Un nome di variabile usato nella definizione di una funzione può apparire o meno nell'elenco degli argomenti. Nel primo caso, il valore del parametro viene specificato quando viene richiamata la funzione. Altrimenti viene usato il valore corrente della variabile del programma.

Le variabili comprese nella lista degli argomenti corrispondono univocamente alle variabili o ai valori degli argomenti che devono essere specificati nella chiamata alla funzione.

Le funzioni definite dall'utente possono essere di tipo numerico o stringa. Il tipo delle funzioni viene specificato dal parametro *name*. Il tipo dell'espressione deve essere coerente con quello della funzione, altrimenti si verifica l'errore "Type mismatch". Se la funzione è di tipo numerico, il risultato dell'espressione viene convertito a questo tipo prima che venga restituito il valore alla funzione.

Se non viene eseguita un'istruzione DEF FN prima che la funzione che essa definisce venga richiamata, si verifica l'errore "Undefined user function".

Esempio

```
  .
  .
  .
400 R=1:S=2
410 DEF FNAB(X,Y)=X^3/Y^2
420 T=FNAB(R,S)
  .
  .
  .
```

La linea 410 definisce la funzione FNAB. La funzione viene richiamata nella linea 420. Dopo l'esecuzione la variabile T contiene il valore (R ^ 3) diviso per (S ^ 2), o .25.

DEF SEG Istruzione

Definisce l'indirizzo iniziale del segmento di memoria corrente. L'istruzione DEF SEG viene generalmente usata all'interno di un programma.

DEF SEG [=*address*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
address	È un'espressione numerica che restituisce un intero senza segno nell'intervallo

ELEMENTO DI SINTASSI	SIGNIFICATO
	da 0 a 65535. L'indirizzo specificato identifica l'indirizzo iniziale del segmento usato da BLOAD, BSAVE, PEEK, POKE, DEF USR e CALL.

Caratteristiche

SE...	ALLORA...
address viene omesso	viene assunto come segmento corrente il Segmento Dati GW-BASIC. Questo è il valore di default iniziale.
address viene specificato	il valore binario viene spostato a sinistra di 4 bit (cioé, se *address* è espresso in esadecimale, gli viene aggiunto uno zero) per ottenere l'indirizzo iniziale del segmento corrente.
viene introdotto un valore fuori dall'intervallo 0-65535	si verifica l'errore "Illegal function call". Il valore precedente viene conservato. GW-BASIC effettua solo questo controllo di tipo formale. Siccome qualsiasi indirizzo di memoria è considerato valido, è responsabilità dell'utente scegliere un valore corretto di *address*, in modo da non richiamare sottoprogrammi ad un indirizzo errato o scrivere su aree contenenti informazioni importanti.

SE...	ALLORA...
DEF e SEG non vengono separate da almeno uno spazio	GW-BASIC interpreta DEFSEG come nome di una variabile. Per esempio: 100 DEFSEG = 150 assegna il valore 150 alla variabile DEFSEG.

L'istruzione DEF SEG può essere usata per allocare sottoprogrammi in linguaggio macchina fuori dal Segmento Dati GW-BASIC. Per far ciò è sufficiente conoscere l'indirizzo iniziale del Segmento Dati e sommarvi 64K. Tale indirizzo è contenuto nella parola situata all'indirizzo assoluto 0:510 Hex.

Esempi

```
10 DEF SEG = &HB800
```

Il segmento corrente è il buffer di video

```
100 DEF SEG
```

Ripristina il Segmento Dati GW-BASIC corrente come segmento.

Si noti che nell'istruzione 10 il buffer di video si trova all'indirizzo assoluto B8000 esadecimale dal momento che l'ultima cifra esadecimale viene omessa nell'istruzione DEF SEG.

DEF USR Istruzione

Abilita l'accesso a un sottoprogramma in linguaggio macchina specificando l'indirizzo di partenza. Il sottoprogramma può essere poi richiamato tramite la funzione USR associata.

L'istruzione DEF USR viene generalmente usata all'interno di un programma.

DEF USR[*n*]=*offset*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	Può essere qualsiasi cifra da 0 a 9. La cifra corrisponde al numero della routine USR il cui indirizzo viene specificato. Se *n* viene omesso, viene assunto DEF USR0.
offset	È un'espressione intera, compresa tra 0 e 65535, che specifica l'indirizzo di partenza del sottoprogramma come spiazzamento rispetto all'indirizzo iniziale del segmento corrente. Il segmento corrente è quello definito dall'ultima istruzione DEF SEG eseguita o coincide con il Segmento Dati GW-BASIC (se nessuna istruzione DEF SEG è stata eseguita).

Caratteristiche

In un programma può essere usato un numero qualsiasi di istruzioni DEF USR per ridefinire gli indirizzi di partenza dei sottoprogrammi, permettendo così l'accesso a un numero illimitato di sottoprogrammi. Per ottenere l'indirizzo di partenza di un sottoprogramma GW-BASIC aggiunge il valore di *offset* al valore dell'indirizzo iniziale del segmento corrente.

Esempio

```
100 DEF SEG=0
 .
 .
 .
200 DEF USR0=24000
210 X=USR0 (Y ^ 2/2.89)
 .
 .
 .
```

DEFINT, DEFSNG, DEFDBL, DEFSTR Istruzioni

Dichiarano il tipo di variabile conformemente alle lettere specificate. Queste istruzioni vengono generalmente usate all'interno di un programma.

DEF*type letter*[-*letter*]][,*letter*[-*letter*]]...

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
type	Può essere INT, SNG, DBL o STR. Non deve essere digitato nessuno spazio tra DEF e INT, SNG, DBL o STR.
letter	Rappresenta una lettera dell'alfabeto (da A a Z).

Caratteristiche

Vengono presi in considerazione tutti i nomi di variabile che iniziano con una o più lettere fra quelle specificate.

Il tipo di variabile specificato da *type* (%, !, #, $) ha sempre la precedenza su un'istruzione DEF*type*. Se non vi sono istruzioni di definizione GW-BASIC considera tutte le variabili senza caratteri di dichiarazione come variabili in semplice precisione. Le istruzioni DEF*type* devono precedere l'uso delle variabili di cui ne definiscono il tipo.

Esempi

```
10 DEFDBL L-P
```

Tutte le variabili che iniziano con le lettere L, M, N, O e P sono variabili in doppia precisione.

```
10 DEFSTR A
```

Tutte le variabili che iniziano con la lettera A sono variabili stringa.

```
10 DEFINT I-N,W-Z
```

Tutte le variabili che iniziano con le lettere I, J, K, L, M, N, W, X, Y, Z sono variabili intere.

DELETE Comando

Cancella linee di programma. Il comando DELETE viene generalmente usato in modo immediato.

DELETE [*linenum1*][-[*linenum2*]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
linenum1	È il numero della prima linea che deve essere cancellata.
linenum2	È il numero dell'ultima linea che deve essere cancellata.

Caratteristiche

GW-BASIC ritorna sempre in stato comandi dopo l'esecuzione di un comando DELETE. Se entrambi *linenum1* e *linenum2* non vengono specificati, si verifica l'errore "Illegal function call". Per indicare la linea corrente può venire usato un punto (.) invece del numero di linea.

Esempi

DELETE 80

Viene cancellata la linea 80.

DELETE 80-120

Vengono cancellate le linee da 80 a 120 comprese.

DELETE 80-

Vengono cancellate tutte le linee a partire dalla linea 80 fino alla fine del programma.

DELETE -80

Vengono cancellate tutte le linee fino alla linea 80 compresa.

DIM Istruzione

Specifica il nome di una matrice, il numero delle sue dimensioni e il massimo valore dell'indice per ogni dimensione. L'istruzione DIM può specificare una o più matrici e deve essere usata all'interno di un programma.

DIM *array(list-of-subscripts)*[,*array(list-of-subscripts)*]...

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
array	È un nome di matrice. Può essere usato un qualsiasi nome di variabile.
list-of-subscripts	È una lista di espressioni numeriche che specificano le dimensioni della matrice. Ogni indice deve essere separato dal successivo con una virgola. Il numero di indici specifica il numero delle dimensioni e il valore di ciascun indice determina il numero di elementi per ogni dimensione.

Caratteristiche

Se viene usato un nome di matrice senza un'istruzione DIM corrispondente, il massimo valore degli indici della matrice è 10. Se viene usato un indice maggiore di quello massimo definito, viene visualizzato il messaggio di errore "Subscript out of range". Il valore minimo di un indice è 0, se l'istruzione OPTION BASE non ha stabilito un valore minimo pari a 1.

Se non viene specificata nessuna istruzione DIM, il primo riferimento ad un elemento di matrice all'interno del programma crea una matrice con il numero di dimensioni specificate. Per esempio, se un'istruzione di programma fa riferimento a:

AR1(3,5,10)

Viene creata la matrice AR1 a 3 dimensioni, e il valore massimo definito per ogni dimensione è 10.

L'istruzione DIM inizializza a zero gli elementi di una matrice di tipo numerico e a stringa nulla gli elementi di una matrice di tipo stringa.

In teoria, il massimo numero di dimensioni consentite in un'istruzione DIM è 255 e il massimo numero di elementi per dimensione è 32767. In pratica, questi valori massimi dipendono sia dalla lunghezza di una linea GW-BASIC, sia dalla dimensione della memoria.

Il tentativo di definire le dimensioni di una matrice più di una volta senza prima averla cancellata provoca la segnalazione d'errore "Duplicate definition". È necessario usare l'istruzione ERASE per cancellare una matrice prima di ridimensionarla. Viene visualizzato lo stesso messaggio di errore se l'istruzione DIM viene preceduta dal riferimento ad una matrice.

Numero di Elementi per Dimensione

SE...	E SE...	ALLORA...
non viene usata l'istruzione DIM	il valore di OPTION BASE è 0	sono consentiti 11 elementi per ogni dimensione (indici da 0 a 10).
	il valore di OPTION BASE è 1	sono consentiti 10 elementi per ogni dimensione (indici da 1 a 10).
viene usata l'istruzione DIM	il valore di OPTION BASE è 0	il numero di elementi per ogni dimensione è ottenuto aggiungendo 1 al valore di ogni indice specificato con DIM.

SE...	E SE...	ALLORA...
	il valore di OPTION BASE è 1	il numero di elementi per ogni dimensione concide con il valore di ogni indice specificato con DIM.

Per Definire una Matrice

L'UTENTE DEVE...	E...	OPPURE...
stabilire il valore minimo dell'indice	può utilizzare l'istruzione OPTION BASE 1	può adottare il valore di default OPTION BASE 0.
assegnare un nome alla matrice	può usare l'istruzione DIM	può far riferimento a un elemento di matrice nel programma.
stabilire il numero di dimensioni	può usare l'istruzione DIM	può far riferimento a un elemento di matrice nel programma.
stabilisce il valore massimo dell'indice per ogni dimensione	può usare l'istruzione DIM	può far riferimento a un elemento di matrice nel programma. In questo caso il numero di elementi per ogni dimensione è uguale a 10.

Osservazioni

Un'istruzione DIM non può stabilire il numero di elementi per ogni dimensione, se l'istruzione viene saltata.

Esempio 1

```
10 DIM A(5),B$(20,30,15)
```

Esempio 2

```
10 INPUT I
20 DIM ARRAY1(I)
30 FOR K=0 TO I
40 READ ARRAY1(K)
50 NEXT K
.
.
.
```

Esempio 3

```
10 DIM A(20)
20 FOR I=1 TO 20
30 READ A(I)
40 NEXT I
.
.
.
```

Esempio 4

```
LIST
10 I=1
20 GOTO 40
30 DIM A(50)
40 A(10)=3
50 A(11)=45
Ok
RUN
Subscript out of range in 50
Ok
```

Il sistema visualizza:

Subscript out of range in 50

quando viene eseguita l'istruzione numero 50, poiché l'istruzione 30 è stata saltata e come massimo valore dell'indice viene assuto per default 10.

DRAW Istruzione

Traccia una figura utilizzando sequenze di comandi del Graphics Macro Language (solo modo grafica).

DRAW *stringexp*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
stringexp	È un'espressione stringa che definisce la sequenza di comandi del Graphics Macro Language (GML) che tracciano la figura.

Caratteristiche

L'istruzione DRAW consente di utilizzare un insieme di comandi di grafica per facilitare la visualizzazione di figure complesse.

Un comando GML è rappresentato da un singolo carattere (per esempio, U, D, L, R, E, F, G, H, M, B, N, A, C, S, X, P) o da una coppia di caratteri (TA) all'interno della stringa *stringexp*, eventualmente seguiti da uno o due valori interi che rappresentano le coordinate di un punto (in pixel).

Un'istruzione WINDOW non ha effetto sull'istruzione DRAW poiché quest'ultima disegna direttamente sulla finestra di grafica in coordinate fisiche.

Comandi di Spostamento

I seguenti comandi di spostamento iniziano il movimento dalla posizione del punto corrente, che di solito coincide con la coordinata dell'ultimo punto tracciato tramite un precedente comando GML (oppure tramite LINE o PSET). Il valore di default della posizione corrente coincide con il centro del video, quando viene attivata l'esecuzione di un programma.

I comandi di spostamento GML specificano i movimenti tramite un intero (*n*) o tramite una coppia di interi (*x*,*y*) nel comando M.

L' effettiva distanza percorsa su video (espressa in pixel) viene data da questo intero moltiplicato per il fattore di scala, definito tramite il comando S.

COMANDO	AZIONE
U [*n*]	Spostamento in alto.
D [*n*]	Spostamento in basso.
L [*n*]	Spostamento a sinistra.
R [*n*]	Spostamento a destra.

COMANDO	AZIONE
E [*n*]	Spostamento diagonale in alto a destra.
F [*n*]	Spostamento diagonale in basso a destra.
G [*n*]	Spostamento diagonale in basso a sinistra.
H [*n*]	Spostamento diagonale in alto a sinistra.
M *x*,*y*	Spostamento assoluto o relativo. Se *x* è preceduto dal segno (+) o meno (—), *x* e *y* vengono sommate algebricamente alle coordinate del punto corrente (movimento relativo). Altrimenti *x*,*y* sono coordinate assolute rispetto all'origine degli assi (movimento assoluto).

Comandi di Prefisso

COMANDO	AZIONE
B	Spostamento senza disegnare. Uno qualunque dei comandi descritti può essere preceduto da B.

COMANDO	AZIONE
N	Spostamento con ritorno alla posizione iniziale. Uno qualunque dei comandi descritti può essere preceduto da N.

Altri Comandi GML

COMANDO	AZIONE
A *n*	Definisce un angolo *n*. *n* può variare fra 0 e 3, dove 0 equivale a 0 gradi, 1 a 90, 2 a 180 e 3 a 270.
TA *n*	Fa ruotare l'angolo *n*. *n* è espresso in gradi, nell'intervallo da -360 a 360. Se *n* è positivo, la rotazione avviene in senso antiorario, se *n* è negativo essa avviene in senso orario. Se *n* si trova fuori dall'intervallo specificato, si verifica l'errore "Illegal function call".
C *n*	Specifica il numero di colore *n*. *n* può variare tra 0 e 3 nel modo a Media Risoluzione, e fra 0 e 1 nei modi ad Alta e Altissima Risoluzione.
S *k*	Definisce il fattore di scala. *k* può variare fra 1 e 255. Il fattore di scala è dato da *k* diviso 4. Il fattore di scala moltiplicato per le

COMANDO	AZIONE
	distanze specificate dai comandi U, D, L, R, E, F, G, H o M, dà lo spostamento effettivo (espresso in pixel). Se viene omesso il comando S viene assunto un valore di scala uguale a 1 (cioé $k=4$).
X *stringexp*	Esegue una sottostringa. Questo comando è molto potente e permette di eseguire una stringa all'interno di un'altra. Si può richiedere che all'interno di una stringa ne venga eseguita una seconda, che a sua volta ne esegue una terza e così via. In *stringexp* gli spazi vengono ignorati; si possono inserire per facilitare la leggibilità.
P *n, m*	*n* rappresenta il numero di colore scelto per colorare l'interno della figura e *m* il numero di colore del contorno. È necessario specificare entrambi i parametri per evitare che si verifichi un errore. Entrambi i parametri possono variare da 0 a 3 nel modo a Media Risoluzione e da 0 a 1 nel modo ad Alta e Altissima Risoluzione.

Osservazioni

In tutti i comandi GML gli argomenti numerici possono essere delle costanti come "327" oppure uguali a *numvar*, dove *numvar* è il nome di una variabile numerica. In questo caso, il nome di variabile deve essere seguito da un punto e virgola (;). Ciò è richiesto anche dal comando X, negli altri casi il punto e virgola per separare i comandi è opzionale.

Su video il numero e la distanza dei pixel lungo gli assi X e Y sono diversi, quindi per avere un segmento di ugual misura lungo i due assi è necessario moltiplicare il numero dei punti verticali per "aspect ratio" (frazione numerica che determina la distanza tra i punti su video) il cui valore per video standard è 4/3. Tali considerazioni devono essere tenute presenti per mantenere la proporzione tra le linee verticali ed orizzontali di una figura.

Il programma seguente traccia un quadrato in Media Risoluzione:

```
10 SCREEN 1
20 A=40
30 B=48
40 DRAW "U=A; R=B; D=A; L=B;"
```

Infatti, in Media Risoluzione ci sono 320 pixel orizzontali e 200 verticali, quindi 16 punti orizzontali sono uguali in lunghezza a 10*4/3 punti verticali e di conseguenza un segmento di 48 punti orizzontali è uguale ad un segmento di 40 punti verticali.

Esempio 1

```
10 U$="U30;" : D$="D30;" : L$="L40;" : R$="R40;"
20 BOX$=U$+R$+D$+L$
30 DRAW "XBOX$;"
40 REM DRAW "XU$;XR$;XD$;XL$;" 'viene tracciato lo stesso
   rettangolo
```

Esempio 2

```
10 DRAW "U50R50D50L50" 'Traccia un rettangolo
20 DRAW "BE 10" 'Si sposta in alto a destra all'interno
                del rettangolo
30 DRAW "P1,3" 'Colora l'interno
```

Esempio 3

```
10 FOR D=0 TO 360 'Traccia alcuni raggi
20 DRAW "TA=D;NU50"
30 NEXT D
```

EDIT Comando

Permette di modificare una linea di programma. Il comando EDIT può essere usato solo in modo immediato.

EDIT { *linenum* | **.** }

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
linenum	È il numero di una linea esistente nel programma. Se la linea specificata non esiste, viene visualizzato il messaggio di errore "Undefined line number".
	Il punto specifica la linea corrente e può essere usato come alternativa a un numero di linea.

Caratteristiche

Quando viene digitato il comando EDIT, GW-BASIC visualizza la linea specificata e posiziona il cursore sotto la prima cifra del numero di linea. La linea può essere quindi modificata tramite i tasti speciali di editor.

Per modificare una linea appena introdotta, si usi il comando EDIT che visualizza di nuovo la linea (in questo caso può essere specificato il simbolo di numero di linea ".").

Per visualizzare un gruppo di linee di programma da modificare, può essere usato anche il comando LIST.

Esempio

Si faccia riferimento al Capitolo 2.

END Istruzione

Termina l'esecuzione di un programma, chiude tutti i file dati aperti e ritorna in stato comandi. L'istruzione END può essere usata solo all'interno di un programma.

END

Caratteristiche

L'istruzione END può essere inserita in un qualunque punto del programma per terminare la sua esecuzione. Al contrario dell'istruzione STOP, l'istruzione END non provoca l'emissione del messaggio "Break in line *nnnn*". L'istruzione END al termine del programma è opzionale. Dopo l'esecuzione di tale istruzione, GW-BASIC ritorna sempre in stato comandi.

Esempio

```
520 IF K>1000 THEN END ELSE GOTO 20
```

ENVIRON Istruzione

Permette di modificare i parametri della Tabella di Definizione dell'Ambiente GW-BASIC.

ENVIRON *parm*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
parm	È una espressione stringa valida che contiene il nuovo parametro stringa che definisce l'ambiente. L'espressione stringa deve contenere solo lettere maiuscole.

Caratteristiche

L'istruzione ENVIRON può essere usata, per esempio, per cambiare il parametro "PATH" in un processo "child".

I parametri possono anche essere trasferiti in un processo "child" con la creazione di un nuovo parametro di ambiente.

Osservazioni

1. *parm* deve avere la forma *parm-id* = *text* dove:

   a. *parm-id* è il nome del parametro, ad esempio "PATH".

b. *parm-id* deve essere separato da *text* con " = " (ad esempio "PATH = ") oppure da spazio. ENVIRON considera tutto quello che si trova alla sinistra del primo spazio oppure di " = " come *parm-id* e tutto quello che si trova a destra come *text*.

c. *text* è il nuovo testo di parametro. Se *text* è una stringa nulla o è formato solo da ";" (un solo punto e virgola, come "PATH = ;") allora il parametro (incluso *parm-id* =) viene tolto dalla Tabella di Definizione dell'Ambiente e la Tabella si riduce.

2. Se nella Tabella di Definizione dell'Ambiente non esiste *parm-id*, allora esso viene aggiunto alla fine di essa.

3. Se *parm-id* esiste allora viene cancellato, la Tabella di Definizione dell'Ambiente si riduce ed un nuovo *parm-id* viene aggiunto alla fine.

Esempi

Il seguente comando di MS-DOS crea un "PATH" di default per la directory root sul disco A:

PATH = A:

PATH può essere mutato in un nuovo valore tramite:

ENVIRON "PATH = A:SALES;A:ACCOUNTS"

Un nuovo parametro può essere aggiunto alla Tabella di Definizione dell'Ambiente:

ENVIRON "SESAME = PLAN"

La Tabella ora contiene:

```
PATH = A:SALES;A:ACCOUNTS
SESAME = PLAN
```

Se poi viene inserito:

ENVIRON "SESAME = ;"

Allora, si deve cancellare SESAME e di conseguenza avere una tabella che contiene:

PATH = A:SALES;A:ACCOUNTS

Errori Possibili

"Type mismatch": *parm* non è una stringa.

"Out of memory": la tabella di Definizione dell'Ambiente risulta piena e nessun altro elemento può venire allocato.

ENVIRON$ Funzione

Permette di estrarre la stringa per l'ambiente specificata dalla Tabella di Definizione dell'Ambiente di GW-BASIC.

ENVIRON$ { **(***parm***)** | **(***nth-parm***)** }

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
parm	È un'espressione stringa che contiene il parametro che deve essere estratto. L'espressione stringa deve contenere solo lettere maiuscole.
nth-parm	È un'espressione intera il cui risultato è un valore compreso tra 1 e 255.

Osservazioni

1. Se viene usato un argomento stringa, la funzione ENVIRON$ restituisce una stringa contenente il testo che segue *parm* = .

2. Se *parm* non viene rintracciato, o non c'è alcun testo che lo segue, allora viene restituita una stringa nulla.

3. Se viene usato un argomento numerico, la funzione ENVIRON$ restituisce una stringa che contiene *nth-parm*, dalla tabella comprendente *parm.*

4. Se non viene definito *nth-parm*, viene restituita una stringa nulla.

Errori Possibili

"Illegal function call": *nth-parm* fuori dall'intervallo numerico.

"Type mismatch": *parm* non è una stringa.

"String too long": la stringa supera i 255 caratteri.

EOF Funzione

Indica la condizione di "end of file". La funzione EOF può essere usata solo all'interno di un programma.

EOF(*filenum*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filenum	È il numero del file specificato nell'istruzione OPEN.

Caratteristiche

Per file sequenziali la funzione EOF restituisce il valore -1 (vero) se viene raggiunta la condizione di "end of file". È consigliabile usare la funzione EOF nelle operazioni di input, per evitare l'errore "Input past end".

Se la funzione EOF viene usata con file ad accesso diretto restituisce "vero" se l'ultima istruzione GET eseguita non era in grado di leggere un intero record, in seguito ad un tentativo di lettura oltre la fine.

Se la funzione EOF viene usata con un device di comunicazione, la definizione della condizione di "end of file" dipende dal modo (ASCII o binario) in cui il device è stato aperto. In modo binario, EOF restituisce "vero" se la coda di input è vuota (LOC (*n*) = 0), restituisce "falso" se la coda di input non è vuota. In modo ASCII, EOF restituisce "falso" finché non viene ricevuto CTRL Z e da quel momento in poi la funzione restituisce "vero" finché il device non viene chiuso.

Esempio

```
 5 DIM M(500)
10 OPEN "I",1, "DATA"
20 K=0
30 IF EOF(1) THEN 100
40 INPUT #1,M(K)
50 K=K+1:GOTO 30
 .
 .
 .
```

Questo esempio legge dati da un file sequenziale denominato "DATA". I valori letti vengono registrati su un vettore M fino al raggiungimento della condizione di "end of file".

ERASE Istruzione

Libera lo spazio e i nomi di variabile precedentemente riservati a una o più matrici. Dopo l'esecuzione del comando i dati vengono persi e le matrici cancellate dalla memoria.

ERASE *array*[,*array*]...

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
array	È il nome della matrice da cancellare.

Caratteristiche

Dopo essere state cancellate, le matrici possono essere ridimensionate, oppure lo spazio di memoria, che era stato allocato per loro, può essere utilizzato per altri scopi. Se si tenta di ridimensionare una matrice che non è stata ancora cancellata, si verifica l'errore "Duplicate definition".

Non è una buona regola di programmazione riutilizzare un identificatore in quanto ciò può generare errori o rendere il programma poco leggibile. Può essere utile ridefinire con lo stesso nome una matrice precedentemente cancellata, per esempio quando si vuole passare più matrici, con numero di dimensioni diverso, ad una stessa subroutine, utilizzando un solo nome di matrice.

Esempio

```
10 DIM A(15,15),B(10,20)
 .
 .
 .
100 ERASE A,B
110 DIM A(100),B(2,2,2)
```

Dopo l'esecuzione dell'istruzione 100, le matrici A e B vengono cancellate e lo spazio di memoria da esse occupato viene reso disponibile. Possono essere definite nuove matrici (istruzione 110) aventi gli stessi nomi ma numero di dimensioni e indici diversi.

ERDEV e ERDEV$ Funzioni

ERDEV è una funzione di tipo intero che contiene il codice di errore restituito dall'ultimo device che ha generato l'errore.

ERDEV$ è una funzione di tipo stringa che contiene il nome del driver del device che ha generato l'errore.

{ **ERDEV | ERDEV$** }

Caratteristiche

ERDEV viene impostata dall'handler Interrupt X'24', quando all'interno di MS-DOS viene rilevato un errore. Negli 8 bit meno significativi ERDEV contiene il codice di errore INT 24, negli 8 bit più significativi contiene i "Word attribute bits"del device headerblock. (b15-b13).

Se l'errore si trova su un Character Device ERDEV$ contiene un nome di device, lungo 8 byte. Se l'errore non si trova su tale device ERDEV$ contiene un nome di device composto da due blocchi di caratteri (A:, B:, C: ecc.).

Osservazioni

Per favorire la compatibilità tra due diverse release è opportuno effettuare il controllo di errori usando ERDEV piuttosto di ERDEV$.

Esempio

Se il driver di device "MYLPT2", installato dall'utente, ha provocato l'errore "Printer out of paper", tramite INT 24, e il numero di

errore del driver per quel problema è 9, ERDEV contiene l'errore numero 9, ERDEV$ contiene "MYLPT2".

ERL ed ERR Funzioni

La funzione ERR restituisce il codice di errore, mentre la funzione ERL restituisce il numero della linea contenente l'errore.

{ **ERL** | **ERR** }

Caratteristiche

Se viene inserita una routine di gestione degli errori, la funzione ERR contiene il codice e la funzione ERL contiene il numero della linea contente l'errore.

Le funzioni ERR e ERL vengono di solito usate in istruzioni IF...THEN per trasferire il controllo del programma alla routine di gestione degli errori.

Se l'istruzione che ha provocato l'errore è un'istruzione di tipo immediato, ERL contiene il numero 65535.

Se non vi è alcun numero di linea a destra di un operatore di confronto la linea non può essere rinumerata tramite il comando RENUM. Poiché ERL e ERR sono funzioni riservate, nessuna delle due può essere posta alla sinistra del segno di uguale in una istruzione LET.

I codici di errore di GW-BASIC sono elencati nell'Appendice A.

Per controllare se si è verificato un errore in una istruzione di tipo immediato, deve essere usata l'istruzione:

```
IF 65535 = ERL THEN...
```

Altrimenti, si usi

IF ERR = *error code* THEN...

IF ERL = *linenum* THEN...

Esempio

```
LIST
10 REM RECTANGLE2
20 ON ERROR GOTO 70
30 INPUT "Length and Width";L,W
40 IF (L<0) OR (W<0) THEN ERROR 200
50 PRINT "Area=";L*W;" L=";L;" W=";W
60 GOTO 30
70 IF (ERR=200) AND (ERL=40)
   THEN PRINT "L or W<0": RESUME 30
80 ON ERROR GOTO 0
90 END
Ok
RUN
Length and Width? -2,5
L or W<0
Length and Width? 2,5
Area= 10 L= 2 W= 5
Length and Width? ^C
Break in 30
Ok
```

Se viene digitato un numero negativo per L o W, viene attivata la routine di gestione degli errori:

```
L o W<0
```

L'esecuzione del programma riprende dall'istruzione 30 (vedere l'istruzione RESUME). Si noti l'uso delle funzioni ERR e ERL nella routine di gestione degli errori.

ERROR Istruzione

Simula il verificarsi di un errore GW-BASIC, oppure genera un errore definito dall'utente.

ERROR *n*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È un'espressione di tipo intero che rappresenta un codice di errore. Deve essere maggiore di 0 e minore o uguale a 255. Se il risultato non è un intero, esso viene arrotondato all'intero più vicino.

Caratteristiche

SE...	ALLORA...
il valore di *n* è uguale a un codice d'errore già utilizzato da GW-BASIC	l'istruzione ERROR simula il verificarsi di tale errore e viene visualizzato il corrispondente messaggio di errore.

SE...	ALLORA...
	Per esempio: **LIST** 10 S=10 20 T=5 30 ERROR S+T 40 END Ok **RUN** String too long in line 30 Ok Oppure, in modo immediato: **ERROR 15** String too long Ok.
il valore di *n* è maggiore di tutti i codici di errore utilizzati da GW-BASIC	l'istruzione ERROR genera un codice d'errore definito dall'utente, che può essere gestito dalla routine di gestione degli errori (vedere l'istruzione ON ERROR in questo capitolo). Nota: un errore è definito dall'utente se viene usato un valore maggiore di tutti i codici di errore utilizzati da GW-BASIC. (È preferibile usare valori molto alti, in modo da mantenere la compatibilità se vengono aggiunti nuovi codice di errore a GW-BASIC).

SE...	ALLORA...
l'istruzione ERROR specifica un codice a cui non è associato alcun messaggio d'errore.	GW-BASIC risponde con il messaggio: "Unprintable error".

EXP Funzione

Eleva *e* (base dei logaritmi naturali) ad una potenza pari al valore dell'argomento.

EXP(*numexp*)

Caratteristiche

numexp deve essere < = 87.3365. Se si verifica una condizione di overflow viene visualizzato il messaggio "Overflow", il risultato è l'infinito di macchina con il segno appropriato e l'esecuzione continua.

La funzione EXP viene calcolata in semplice precisione, se non viene impostato /D nella linea di comando GWBASIC.

Esempio

```
10 X=5
20 PRINT EXP(X-1)
RUN
 54.59815
Ok
```

FIELD Istruzione

Alloca lo spazio per le variabili nel buffer di un file ad accesso diretto. L'istruzione FIELD viene generalmente usata all'interno di un programma.

FIELD [#]*filenum*,*width* **AS** *stringvar* [,*width* **AS** *stringvar*]...

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filenum	È il numero con cui il file è stato aperto (OPEN).
width	È il numero di caratteri da riservare a *stringvar*.
stringvar	È un nome di variabile stringa che deve essere usato per accedere ad un file ad accesso diretto.

Caratteristiche

Prima dell'esecuzione di un'istruzione GET o PUT deve essere eseguita l'istruzione FIELD, per definire il formato del buffer per file ad accesso diretto.

Il numero complessivo dei byte allocati da un'istruzione FIELD non deve superare la lunghezza del record specificata all'atto dell'apertura del file. In caso contrario si verifica l'errore "Field overflow". (La lunghezza di default per il record è 128 byte.)

Su uno stesso file può essere eseguito un numero qualsiasi di istruzioni FIELD e tutte sono contemporaneamente valide.

Osservazioni

Non si deve usare un nome di variabile definita tramite l'istruzione FIELD in un'istruzione di input o alla sinistra del segno di uguale in un'istruzione di assegnazione.

Dopo che un nome di variabile è stato definito tramite l'istruzione FIELD, esso punta all'indirizzo corretto nel buffer del file ad accesso diretto. Se vengono eseguite successivamente le istruzioni INPUT o LET, con il nome di variabile a sinistra del segno di uguale, la variabile non fa più riferimento al buffer del file ad accesso diretto, ma alle variabili memorizzate nello spazio stringhe.

Si noti che un nome di una variabile, definita precedentemente in un'istruzione FIELD, può essere inserita a destra del segno di uguale in un'istruzione di assegnazione.

Esempio 1

```
10 FIELD 1,20 AS N$,10 AS ID$,40 AS ADD$
```

Riserva le prime 20 posizioni (byte), in un buffer di un file ad accesso diretto, alla variabile stringa N$, le successive 10 posizioni a ID$ e le successive 40 posizioni ad ADD$. FIELD non pone nessun dato nel buffer del file ad accesso diretto.

Esempio 2

```
10 OPEN "R",#1,"A:PHONELST",35
15 FIELD #1,2 AS RECNBR$,33 AS DUMMY$
20 FIELD #1,25 AS NAME$,10 AS PHONENBR$
25 GET #1
30 TOTAL=CVI(RECNBR$)
35 FOR I=2 TO TOTAL
40 GET #1, I
45 PRINT NAME$, PHONENBR$
50 NEXT I
```

Illustra una definizione multipla dell'istruzione FIELD. Nell'istruzione 15, è definito un campo di 35 byte per il primo record allo scopo di conservare il numero dei record nel file; l'istruzione 20 definisce il campo per ogni nominativo di persona e numero di telefono.

Esempio 3

```
10 FOR LOOP%=0 TO 7
20 FIELD #1,(LOOP%*16) AS OFFSET$,16 AS A$(LOOP%)
30 NEXT LOOP%
```

Mostra la costruzione di un'istruzione FIELD che usa un vettore di elementi tutti della stessa dimensione. Il risultato è equivalente ad un'unica dichiarazione:

```
FIELD #1,16 AS A$(0),16 AS A$(1),...,16 AS A$(6),16 AS A$(7)
```

Esempio 4

```
10 DIM SIZE% (4%): REM ARRAY DELLE DIMENSIONI DEL
   CAMPO
20 FOR LOOP%=0 TO 4%:READ SIZE% (LOOP%):
   NEXT LOOP%
30 DATA 9,10,12,21,41
   .
   .
```

```
120 DIM A$(4%): REM ARRAY DELLE VARIABILI ALLOCATE
130 OFFSET% = 0
140 FOR LOOP% = 0 TO 4%
150 FIELD #1,OFFSET% AS OFFSET$,SIZE%(LOOP%)
    AS A$(LOOP%)
160 OFFSET% = OFFSET% + SIZE%(LOOP%)
170 NEXT LOOP%
```

Viene creato un campo nello stesso modo dell'esempio 3. Comunque la dimensione di ogni elemento varia in funzione dell'elemento stesso. La dichiarazione equivalente è:

```
FIELD #1,SIZE%(0) AS A$(0),SIZE%(1) AS A$(1),...
SIZE%(4%) AS A$(4%)
```

Esempio 5

```
10 FIELD#1,255 AS TST$
```

Si noti che le variabili devono osservare la limitazione della lunghezza massima. Per esempio nell'istruzione FIELD precedente la lunghezza massima di TST$ è 255.

FILES Comando

Visualizza il nome dei file residenti sulla directory specificata.

FILES [*filespec* I *pathname*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filespec (o *pathname*)	È un'espressione stringa che specifica un nome di file o un nome di directory e, opzionalmente, un drive. Se il drive viene omesso, viene assunto il drive di default di MS-DOS. Se non viene specificata una directory, viene assunta la directory corrente per il drive specificato. Se viene specificato solo FILES, vengono visualizzati tutti i file residenti sulla directory corrente del drive di default di MS-DOS.

Caratteristiche

Un nome di file può contenere punti interrogativi (?) o asterischi (*), che vengono usati come caratteri jolly. Un punto interrogativo può sostituire un solo carattere del nome del file o della sua estensione. Un asterisco, invece, può sostituire uno o più caratteri che iniziano in quella posizione. L'asterisco è l'abbreviazione di una serie di punti interrogativi. Non è necessario usare l'asterisco quando vengono richiesti tutti i file su un drive, per esempio FILES "B:".

Esempi

FILES

Visualizza tutti i file residenti sulla directory corrente.

FILES "*.BAS"

Visualizza tutti i file aventi l'estensione .BAS.

FILES "A:*.*"

Visualizza tutti i file residenti sul drive A.

FILES "A:"

Equivale all'esempio precedente.

FILES "GEO?.BAS"

Visualizza tutti i file sulla directory corrente del drive di default di MS-DOS aventi un nome di file di 4 caratteri che inizia con GEO ed ha estensione .BAS.

Le sottodirectory vengono specificate con DIR dopo il nome della directory.

FILES "\SALES"

Se SALES è una sottodirectory della directory corrente, questo comando visualizza SALES DIR. Se SALES è un file della directory corrente, questo comando visualizza SALES.

FILES "\SALES \MARY"

Visualizza MARY DIR se MARY è una sottodirectory di SALES, oppure, se MARY è un file visualizza il suo nome.

FIX Funzione

Calcola la parte intera dell'argomento (troncamento).

FIX(*numexp*)

Caratteristiche

FIX(*numexp*) è equivalente a SNG (*numexp*) *INT (ABS (*numexp*)). La principale differenza tra FIX e INT è costituita dal fatto che FIX non restituisce il valore immediatamente inferiore in caso di argomenti negativi.

Esempi

```
PRINT FIX(58.75)
 58
Ok

PRINT FIX(-58.75)
-58
Ok
```

FOR...NEXT Istruzioni

Consentono di eseguire in ciclo una serie di istruzioni un dato numero di volte.

Le istruzioni FOR/NEXT vengono generalmente usate all'interno di un programma.

FOR *control-variable* = *initial-value* **TO** *final-value*[**STEP** *increment*]

.
.
.
[*loop statements*]
.
.
.
NEXT [*control-variable*][,*control-variable*]...

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
control-variable	È una variabile intera o in semplice precisione da usare come contatore, detta variabile di controllo.
initial-value	È un'espressione numerica che specifica il valore iniziale assegnato alla variabile di controllo.

ELEMENTO DI SINTASSI	SIGNIFICATO
final-value	È un'espressione numerica che rappresenta il valore finale della variabile di controllo.
increment	È un'espressione numerica il cui valore rappresenta l'incremento, cioé il valore da aggiungere (con il suo segno algebrico) alla variabile di controllo ad ogni esecuzione dell'istruzione NEXT.

Caratteristiche

Le linee di programma che seguono l'istruzione FOR vengono eseguite fino a che non viene incontrata l'istruzione NEXT. A questo punto il parametro *control-variable* viene incrementato con il valore specificato da *increment*. Viene quindi verificato se il valore della variabile di controllo è superiore a *final-value*. Se il valore di *control-variable* è inferiore al valore finale, il controllo del programma ritorna all'istruzione che segue FOR e l'elaborazione viene ripetuta. Se il valore di *control-variable* è superiore al valore finale, l'esecuzione continua con l'istruzione successiva all'istruzione NEXT. Questo è un ciclo FOR...NEXT.

Se la clausola STEP non viene specificata, viene assunto l'incremento uguale a uno. Se l'incremento è negativo, il valore finale deve essere minore di quello iniziale. *Control-variable* viene decrementato ad ogni ciclo; la sequenza di istruzioni tra FOR e NEXT viene eseguita ripetutamente finché il valore di *control-variable* diventa minore di *final-value*.

Control-variable deve essere una variabile numerica intera o in semplice precisione. Se viene usata una variabile numerica in doppia precisione si verifica l'errore ''Type mismatch''.

Il ciclo non viene eseguito se:

- l'incremento è positivo e *initial-value* è superiore a *final-value*
- l'incremento è negativo e *initial-value* è inferiore a *final value*.

Cicli Nidificati

I cicli FOR...NEXT possono essere nidificati, cioé un ciclo FOR...NEXT può essere compreso all'interno di un altro ciclo FOR...NEXT. Nei cicli nidificati, ogni ciclo deve avere un proprio nome di variabile di controllo. L'istruzione NEXT per il ciclo interno deve essere eseguita prima di quella per il ciclo esterno. Se i cicli nidificati hanno lo stesso punto finale, è possibile usare per tutti una sola istruzione NEXT (con una lista di variabili di controllo).

Si noti che un'istruzione in questa forma:

```
100 NEXT V1, V2, V3
```

svolge esattamente le stesse funzioni della sequenza di istruzioni:

```
100 NEXT V1
110 NEXT V2
120 NEXT V3
```

Le variabili presenti nell'istruzione NEXT possono essere omesse, nel qual caso l'istruzione NEXT viene associata all'ultima istruzione FOR eseguita. Quando si usano cicli nidificati è necessario specificare le variabili di controllo in ogni istruzione NEXT.

Se viene incontrata un'istruzione NEXT prima della corrispondente istruzione FOR, viene visualizzato il messaggio di errore "NEXT without FOR" e l'esecuzione del programma termina.

Cicli di Ritardo

È possibile, all'interno di un programma, inserire tra due istruzioni un ciclo FOR/NEXT fittizio (cioé senza alcuna istruzione tra FOR e NEXT) per ritardare l'esecuzione della seconda istruzione. Ciò può essere utile in varie occasioni, ad esempio per l'animazione di figure.

Esempio 1

```
10 K=10
20 FOR I=1 TO K STEP 2
30 PRINT I;
40 K=K+10
50 PRINT K
60 NEXT
RUN
 1  20
 3  30
 5  40
 7  50
 9  60
Ok
```

Esempio 2

```
10 J=0
20 FOR I=1 TO J
30 PRINT I
40 NEXT I
```

In questo esempio il ciclo non viene eseguito perché il valore iniziale della variabile di controllo supera il valore finale.

Esempio 3

```
10 I=5
20 FOR I=1 TO I+5
30 PRINT I;
40 NEXT
RUN
 1 2 3 4 5 6 7 8 9 10
Ok
```

In questo esempio il ciclo viene eseguito dieci volte. Il valore finale della variabile di controllo viene sempre calcolato prima di quello iniziale.

FRE Funzione

Fornisce il numero di byte presenti in memoria che non sono utilizzati da GW-BASIC.

FRE(*dummy*)

Caratteristiche

L'argomento di FRE è un argomento fittizio, perciò può essere specificato un qualsiasi valore. FRE (""), prima di calcolare il numero di byte disponibile, effettua l'eliminazione degli spazi vuoti (garbage collection).

GW-BASIC non inizia una "garbage collection" finché non vi è una saturazione della memoria disponibile. Quindi, un uso periodico di FRE ("") abbrevia i tempi di ogni "garbage collection".

Esempio

```
PRINT FRE(0)
 14542
Ok
```

GET (File) Istruzione

Legge un record da un file ad accesso diretto in un buffer ad accesso diretto.

L'istruzione GET (File) viene generalmente usata all'interno di un programma.

GET [#]*filenum*[,*recordnum*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filenum	È il numero con cui il file è stato aperto.
recordnum	È il numero (nell'intervallo da 1 a 16777215) del record da leggere. Se viene omesso, viene letto nel buffer il record successivo (dopo l'ultima istruzione GET).

Caratteristiche

Il numero massimo di record è 16777215. Questo permette la creazione di grossi file formati da record corti.

Dopo l'esecuzione di un'istruzione GET è possibile usare i riferimenti alle variabili definite nell'istruzione FIELD o le istruzioni INPUT#, LINE INPUT# per leggere caratteri dal buffer ad accesso diretto.

Dopo un'istruzione GET è possibile usare la funzione EOF per controllare se GET si trova oltre la fine del file.

Esempio

```
LIST
10 OPEN "R",1,"A:RAND",48
20 FIELD 1,20 AS R1$,20 AS R2$,8 AS R3$
30 FOR L=1 TO 4
40 GET 1,L
50 PRINT R1$,R2$,R3$
60 NEXT
70 CLOSE 1
80 END
Ok
RUN
Super man          Usa            11234621
robin hood         England        23462101
.
.
.
Ok
```

Questo programma reperisce le informazioni memorizzate nel file specificato. I dati trasferiti nel buffer sono accessibili da programma. Ciò avviene in questo caso tramite l'istruzione PRINT (istruzione 50). Questi dati sono stati scritti in precedenza sul file tramite l'istruzione PUT (File).

GET (File COM) Istruzione

Legge un numero specificato di byte dal buffer di comunicazione.

L'istruzione GET (File COM) viene generalmente usata all'interno di un programma.

GET [#]*filenum*, *length*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filenum	È un'espressione intera che restituisce un numero di file.
length	È un'espressione intera che restituisce il numero di byte da trasferire nel buffer di comunicazione.Tale numero non può superare il valore fissato dalla clausola LEN nell'istruzione OPEN COM.

Esempio

```
100 GET #2,80
```

GET (Grafica) Istruzione

Trasferisce in un vettore le immagini di grafica presenti su di un'area rettangolare del video.

GET [STEP] (*x1*, *y1*) - [STEP] (*x2*, *y2*), *array*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
(*x1*, *y1*)-(*x2*, *y2*)	Definisce un'area rettangolare su video. *x1*, *y1* sono le coordinate in alto a sinistra e *x2*, *y2* quelle in basso a destra del rettangolo. Possono essere date in forma assoluta o relativa (se viene usata l'opzione STEP).
array	È il nome assegnato al vettore destinato a contenere l'immagine, delimitata dal rettangolo specificato.

Caratteristiche

L'istruzione GET trasferisce all'interno del vettore l'immagine di video delimitata dal rettangolo descritto dai punti specificati. Questa immagine può essere ritrasferita su video anche in posizioni diverse tramite l'istruzione PUT. GET e PUT permettono l'animazione e il movimento degli oggetti ad alta velocità.

Il vettore deve essere di tipo numerico cioé con elementi interi, oppure in semplice o doppia precisione; deve avere dimensioni sufficientemente grandi per poter contenere l'immagine visualizzata.

Se il vettore non è di tipo intero, il suo contenuto dopo l'esecuzione di GET risulta di difficile interpretazione, per cui è consigliabile usare vettori con elementi interi.

Dimensioni del Vettore

Il formato di memorizzazione nel vettore è il seguente:

2 byte che contengono la dimensione *x* dell'area rettangolare (in pixel).
2 byte che contengono la dimensione *y* dell'area rettangolare (in pixel)
I dati stessi del vettore.

Ogni riga di pixel su video viene scandita a partire dal margine sinistro dell'area rettangolare, dall'alto in basso; negli elementi del vettore vengono memorizzati, di volta in volta, i numeri di colore di 8 pixel consecutivi. Così se è stata memorizzata una riga di pixel inferiore ad un multiplo di 8 l'elemento del vettore, corrispondente all'ultimo pixel della riga, sarà completato con zeri. La dimensione richiesta per il vettore (in byte) è:

---

4 + INT ((*x* * *bitsperpixel* + 7) /8) * *y*

Dove

bitsperpixel è pari a 2 in Media Risoluzione e 1 in Alta e Altissima Risoluzione.

I byte per ogni elemento del vettore sono:

2 se il vettore ha elementi interi
4 se il vettore ha elementi in semplice precisione
8 se il vettore ha elementi in doppia precisione

Osservazioni

Se si vuole traferire (GET) in un vettore con elementi di tipo intero un'immagine di 10 × 12 pixel di lato, in Media Risoluzione, il numero di byte richiesto è 4 + INT ((10* 2 + 7)/8) *12 ovvero 40 byte. Cioé è necessario un vettore con almeno 20 elementi.

Se viene usato un vettore con elementi di tipo intero, è possibile esaminare le dimensioni *x* ed *y* ed anche i dati stessi. La dimensione *x* è nell'elemento 0 del vettore e la dimensione *y* è nell'elemento 1. Bisogna ricordare che gli interi sono memorizzati con il byte meno significativo per primo seguito dal byte più significativo, ma i dati vengono trasferiti, da video in memoria, a partire da sinistra, cioé il byte più significativo per primo seguito da quello meno significativo.

Esempio

```
10 CLS: SCREEN 3: PSET(20,20)
20 X$="R20D20L20U20":DRAW X$
30 DIM BOX%(64):GET(20,20)-(40-40),BOX%
35 PRINT "PRESS A KEY": A$ = INPUT$(1)
40 CLS: PUT(100,100),BOX%
```

GOSUB...RETURN Istruzioni

GOSUB trasferisce il controllo ad un sottoprogramma GW-BASIC saltando alla linea specificata.

RETURN trasferisce il controllo all'ultima istruzione che segue GOSUB (o ON...GOSUB) o ad una linea specificata.

GOSUB/RETURN vengono usate solo all'interno di un programma.

GOSUB *linenum1*

.
.
.

RETURN [*linenum2*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
linenum1	È la prima linea del sottoprogramma.
linenum2	È qualsiasi linea del programma diversa da linenum1 e dal numero di linea dell'istruzione GOSUB.

Caratteristiche

Un sottoprogramma può essere richiamato da un programma per un qualsiasi numero di volte. Un sottoprogramma può venire anche richiamato all'interno di un altro programma. Tale concatenamento di sottoprogrammi viene limitato solo dalla memoria disponibile.

L'istruzione RETURN in un sottoprogramma provoca il ritorno all'istruzione che segue l'istruzione GOSUB o ON...GOSUB eseguita per ultima. Un sottoprogramma può contenere più di un'istruzione RETURN, se la logica del programma lo richiede.

L'opzione *linenum2* può essere inclusa nell'istruzione RETURN per ritornare dal sottoprogramma ad un numero di linea specifico. Si deve usare questo tipo di rientro con attenzione, poiché qualsiasi istruzione GOSUB, WHILE o FOR attiva in quel momento rimane attiva e può generare l'errore "FOR without NEXT".

I sottoprogrammi possono essere situati in qualsiasi punto del programma, ma si raccomanda che il sottoprogramma sia facilmente distinguibile dal programma principale. Per evitare un ingresso involontario nel sottoprogramma, questo deve essere preceduto da un'istruzione STOP, END oppure GOTO che effettua il controllo del sottoprogramma.

Se *linenum1* o il *linenum2* non sono linee del programma, si verifica l'errore "Undefined line number".

Esempio

```
10 GOSUB 40
20 PRINT "BACK FROM SUBROUTINE"
30 END
40 PRINT "SUBROUTINE";
50 PRINT "IN";
60 PRINT "PROGRESS"
70 RETURN
RUN
SUBROUTINE IN PROGRESS
BACK FROM SUBROUTINE
Ok
```

GOTO Istruzione

Trasferisce il controllo alla linea di programma specificata.

GOTO *linenum*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
linenum	È una linea del programma.

Caratteristiche

Se *linenum* si riferisce ad un'istruzione eseguibile, vengono eseguite la stessa e le successive; se si riferisce ad un'istruzione non eseguibile, l'esecuzione viene ripresa dalla prima istruzione eseguibile dopo *linenum*.

Se l'istruzione specificata da *linenum* non esiste nel programma, viene visualizzato il messaggio di errore "Undefined line number".

Esempio

```
LIST
10 READ R
20 PRINT "R = ";R,
30 A = 3.14*R ^ 2
40 PRINT "AREA = ";A
50 GOTO 10
60 DATA 5,7,12
Ok
RUN
R = 5         AREA = 78.5
R = 7         AREA = 153.86
R = 12        AREA = 452.16
?Out of DATA in 10
Ok
```

GWBASIC Comando

Inizializza GW-BASIC e l'ambiente operativo (GWBASIC è un comando di MS-DOS, non un comando di GW-BASIC).

GWBASIC [*filespec* | *pathname*] [<*stdin*] [[>] >*stdout*]
[**/F:** *number-of-files*] [**/S:** *lrecl*] [**/C:** *buffer-size*]
[**/M:** [*highest-memory*] [, *max-blocksize*]] [**/D**] [**/I**]

Dove

Le opzioni che iniziano con una barra (/) si chiamano switch. Uno switch è usato per specificare i parametri.

ELEMENTO DI SINTASSI	SIGNIFICATO
filespec (o pathname)	È una stringa di caratteri (non racchiusa tra doppi apici) che specifica un file di programma GW-BASIC. Se il file è presente, GW-BASIC procede come se venisse dato un comando RUN *filespec* dopo il completamento dell'inizializzazione. Se non viene specificata alcuna estensione e se il filename ha una lunghezza inferiore a 9 caratteri viene usata l'estensione di default .BAS. L'opzione *filespec* permette l'esecuzione in batch di programmi GW-BASIC, inserendo questo tipo di comando nel file AUTOEXEC.BAT. Per terminare i programmi GW-BASIC che vengono eseguiti in questo modo deve essere usato il comando SYSTEM, in modo da permettere l'esecuzione del comando successivo contenuto nel file AUTOEXEC.BAT.
stdin	È una stringa di caratteri (non racchiusa tra doppi apici) per la specificazione del file di input standard. L'input di GW-BASIC viene reindirizzato dal file specificato da *stdin*. Se presente, questo parametro deve comparire prima di

ELEMENTO DI SINTASSI	SIGNIFICATO
	qualsiasi switch (vedere "Reindirizzamento di Input ed Output Standard").
stdout	È una stringa di caratteri (non racchiusa tra doppi apici) per la specificazione del file di output standard. GW-BASIC viene reindirizzato al file specificato da *stdout*. Se presente, questo parametro deve comparire prima di qualsiasi switch (vedere "Reindirizzamento di Input e Output Standard).
/F: *number-of-files*	Questo switch stabilisce il numero massimo di file (da 1 a 15) che possono essere aperti simultaneamente durante l'esecuzione di un programma GW-BASIC. Tale numero viene ignorato se lo switch /I non è specificato sulla linea di comando. Se questo switch e lo switch /I sono presenti, allora il numero massimo di file è pari a *number-of-files*. Ogni file richiede 62 byte per il File Control Block (FCB) più 128 byte per il buffer dati. La dimensione del buffer dati può essere alterata tramite lo switch /S:. Se l'opzione /F: viene omessa, *number-of-file* è pari a 3.

▶

ELEMENTO DI SINTASSI	SIGNIFICATO
	Il numero di file aperti che MS-DOS può gestire dipende dal valore di FILES = *parameter* nel file CONFIG.SYS. Si raccomanda FILE = 10 per GW-BASIC. I primi tre vengono usati da *stdin*, *stdout*, *stderr*, *stdaux* e *stdprn*. GW-BASIC richiede un handler di file supplementare per i comandi LOAD, SAVE, CHAIN, NAME e MERGE. Questo riserva 6 file per le operazioni di I/O così /F:6 è il massimo supportato da MS-DOS quando FILES = 10 compare nel file CONFIG.SYS. Il tentativo di aprire un file dopo che tutti gli handler di file sono stati esauriti genera l'errore "Too many files".
/S:*lrecl*	Questo switch imposta la massima lunghezza di record permessa con i file ad accesso diretto. Esso viene ignorato se sulla linea di comando non è specificato lo switch /I. Se questo switch e lo switch /I sono presenti, la massima lunghezza di record è pari a *lrecl*. L'opzione record-length nell'istruzione OPEN non può superare questo valore. Se l'opzione /S: viene omessa, la lunghezza di record è pari a 128 byte. Il valore massimo permesso per *lrecl* è 32767 byte.

ELEMENTO DI SINTASSI	SIGNIFICATO
/C: *buffer-size*	Se è presente, controlla le comunicazioni RS232. /C:0 disabilita le funzioni RS232; qualsiasi tentativo di I/O successivo genera l'errore "Device unavailable". Con la specificazione di /C:*n*, si allocano *n* byte per il buffer di ricezione di ogni porta RS232 presente. Se l'opzione /C: viene omessa, GW-BASIC alloca 256 byte per il buffer di ricezione di ogni porta RS232. Vengono sempre assegnati 128 byte al buffer di trasmissione. Il valore massimo permesso per *buffer-size* è 32767.
/M:[*highest-memory*] [,*max-blocksize*]	Se presente, *highest-memory* indica il numero massimo di byte a disposizione di GW-BASIC come area di lavoro. GW-BASIC tenta di allocare 64k di memoria per i dati e per il segmento di stack. Se con i programmi GW-BASIC devono essere usati sottoprogrammi in linguaggio macchina è necessario usare lo switch /M:, per stabilire la posizione di memoria massima che può usare GW-BASIC. Quando questa viene omessa o è 0, GW-BASIC tenta di allocare l'area più grande possibile fino ad un massimo di 65536 byte. Per caricare i programmi nell'area di lavoro di GW-BASIC è necessario usare il ▶

ELEMENTO DI SINTASSI	SIGNIFICATO
	parametro opzionale *max-blocksize* in modo da riservare lo spazio per l'area di lavoro ed i programmi utente. Questo è necessario se si intende usare il comando SHELL. Se quest'operazione non viene compiuta correttamente quando viene eseguito un comando SHELL viene caricato all'inizio delle routine utente. Il parametro *max-blocksize* deve essere in Paragrafi (byte multipli di 16). Se viene omesso si assume &H1000 (4096) che alloca 65536 byte (65536 = 4096 × 16) per il segmento dati di GW-BASIC ed il segmento di Stack. Se è necessario avere 65536 byte per GW-BASIC e 512 byte per i sottoprogrammi in linguaggio macchina, allora bisogna usare /M:.&H1010 (4096 paragrafi per GW-BASIC + 16 paragrafi per le routine utente). Questa opzione può essere anche usata per ridurre il blocco di GW-BASIC in modo da liberare più memoria per il caricamento e l'esecuzione di altri programmi. /M:,2048 significa: "Allocare ed usare un massimo di 32768 byte per i dati e lo stack". /M:32000,2048 alloca un massimo di 32768 byte, ma GW-BASIC ne usa solo un numero inferiore a 32000. Questo lascia 768 byte disponibili per i programmi.

ELEMENTO DI SINTASSI	SIGNIFICATO
/D	Se è presente, permette al package matematico in Doppia Precisione Trascendentale di rimanere residente. Le funzioni che vengono calcolate in doppia precisione, se questo package è residente, sono: ATN, COS, EXP, LOG, SIN, SQR e TAN. Se omesso, questo package viene eliminato e lo spazio viene messo a disposizione del programma. La quantità di memoria richiesta da questo package è pari approssimativamente a 3000 byte.
/I	GW-BASIC è in grado di allocare dinamicamente lo spazio necessario per effettuare le operazioni sui file. Per questo motivo GW-BASIC non ha bisogno di utilizzare gli switch /S: e /F:. Alcune applicazioni vengono scritte in modo tale che certe strutture di dati interne di BASIC devono essere statiche. Per garantire la compatibilità con questi programmi BASIC, GW-BASIC alloca staticamente lo spazio necessario per le operazioni di file basate sugli switch /D ed /F: quando lo switch /I viene specificato.

number-of-files, lrecl, buffer-size, highest-memory e max-blocksize possono essere numeri decimali, ottali (preceduti da &O) oppure esadecimali (preceduti da &H).

Esempi

A >**GWBASIC PAYROLL**

Usa 64k di memoria e 3 file, carica ed esegue PAYROLL.BAS.

A >**GWBASIC INVENT/F:6/I**

Usa 64k di memoria e 6 file, carica ed esegue INVENT.BAS.

A >**GWBASIC /C:0/M:32768**

Disabilita il supporto di RS232 ed usa solo i primi 32k di memoria.

A >**GWBASIC /F:4/S:512/I**

Usa 4 file e consente la lughezza massima di record pari a 512 byte.

A >**GWBASIC TTY/C:512**

Usa 64k di memoria e 3 file, alloca 512 byte nei buffer di ricezione RS232, carica ed esegue TTY.BAS.

Reindirizzamento di Input ed Output Standard

Con GW-BASIC è possibile reindirizzare l'input e l'output. Generalmente, l'input standard corrisponde ai caratteri impostati sulla tastiera, ma può esere anche il contenuto di un qualsiasi file specificato sulla linea del comando GWBASIC.

L'output standard, generalmente su video, può essere reindirizzato ad un qualsiasi device o file specificato sulla linea del comando GWBASIC.

Osservazioni

1. Quando sono reindirizzate, tutte le istruzioni INPUT, LINE INPUT, INPUT$ e INKEY$ leggono dal file specificato tramite il parametro *stdin* invece che dalla tastiera.

2. Tutte le istruzioni PRINT e PRINT USING scrivono sul file o sulla periferica specificata tramite il parametro *stdout* invece che sul video. Se vengono introdotti due simboli di maggiore (> >) prima di *stdout*, l'output di PRINT o PRINT USING viene aggiunto alla fine del file specificato.

3. I messaggi di errore sono diretti all'output standard e compaiono su video.

4. Input di file da KYBD: legge anche in questo caso dalla tastiera.

5. Output di file su SCRN: è indirizzato anche in questo caso al video.

6. GW-BASIC continua a rilevare i tasti dalla tastiera quando viene usata l'istruzione ON KEY(*n*).

7. Il tasto di eco della stampante non genera eco di LPT1: se l'Output Standard è stato reindirizzato.

8. Premere **CTRL BREAK** provoca la chiusura da parte di GW-BASIC di qualsiasi file aperto, l'emissione su output standard del messaggio "Break in line nnnnn", l'uscita da GW-BASIC e il rientro a MS-DOS.

9. Quando l'input è reindirizzato GW-BASIC continua a leggere da questo sorgente finché viene rilevato **CTRL Z**. Questa condizione può essere controllata tramite la funzione EOF. Se il file non termina con **CTRL Z** o si tenta di leggere l'end-of-file precedente con un'istruzione INPUT#, tutti i file aperti vengono chiusi. Il messaggio "Read past end" è inviato all'output standard e da GW-BASIC si ritorna a MS-DOS.

10 Considerato il modo in cui MS-DOS gestisce i file testo, non è consigliabile eseguire un programma GW-BASIC in cui l'output viene ridiretto e aggiunto ad un file creato precedentemente (sia con EDLIN sia in modo sequenziale); i comandi come TYPE possono mostrare solo il contenuto originale.

Esempi

GWBASIC MYPROG > DATA.OUT

Dalla tastiera continuano a pervenire i dati letti dalle istruzioni INPUT e LINE INPUT. L'output dei dati, tramite l'istruzione PRINT, è diretto al file DATA.OUT.

GWBASIC MYPROG < DATA.IN

Da DATA.IN provengono i dati letti tramite le istruzioni INPUT e LINE INPUT. L'output dei dati, tramite l'istruzione PRINT, è diretto al video.

GWBASIC MYPROG < MYINPUT.DAT > MYOUTPUT.DAT

Dal file MYINPUT.DAT provengono i dati letti tramite le istruzioni INPUT e LINE INPUT e l'output, tramite l'istruzione PRINT, è diretto al file MYOUTPUT.DAT.

GWBASIC MYPROG <\SALES\JOHN\TRANS.> > \SALES\SALES.DAT

I dati letti dalle istruzioni INPUT e LINE INPUT provengono dal file \SALES \JOHN \TRANS. L'output dei dati viene aggiunto alla fine del file \SALES \SALES.DAT, tramite l'istruzione PRINT.

HEX$ Funzione

Restituisce una stringa che rappresenta il valore esadecimale dell'argomento decimale.

HEX$(*numexp*)

Caratteristiche

Il parametro *numexp* viene arrotondato all'intero prima di calcolare il valore della funzione HEX$. Se *numexp* è negativo viene usata una rappresentazione in complemento a due.

Esempio

```
10 INPUT X
20 A$ = HEX$(X)
30 PRINT X "DECIMAL IS" A$ "HEXADECIMAL"
RUN
? 32
 32 DECIMAL IS 20 HEXADECIMAL
Ok
```

Per dettagli sulla conversione ottale si faccia riferimento alla funzione OCT$.

IF...GOTO...ELSE e IF...THEN...ELSE Istruzioni

In base al risultato di una certa condizione prendono decisioni sul flusso del programma. Le istruzioni IF...GOTO...ELSE ed IF...THEN...ELSE vengono generalmente usate all'interno di un programma.

Sintassi 1

IF *condition* **GOTO** *linenum*
[**ELSE** {*statements* | *linenum*}]

Sintassi 2

IF *condition* **THEN** {*statements* | *linenum*} **[ELSE** {*statements* | *linenum*}]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
condition	Può essere un'espressione numerica, di confronto o logica. GW-BASIC determina se la condizione è vera o falsa, controllando se il suo risultato (numerico) è diverso da zero (condizione falsa). Per questo motivo è possibile verificare se il valore di una variabile è uguale a zero o diverso da zero, specificando semplicemente il nome della variabile come *condition*.
statements	Sono una o più istruzioni. Ogni istruzione deve essere separata da quella precedente da due punti (:).
linenum	È un numero di linea appartenente al programma in memoria.

Caratteristiche

Se il risultato di *condition* è vero (cioé diverso da zero), viene eseguita la clausola GOTO o THEN. GOTO è sempre seguita da un numero di linea; THEN può essere seguita o da un numero di

linea (per saltare ad un altro punto del programma) o da una o più istruzioni eseguibili. Se il risultato di *condition* è falso (cioé uguale a zero) GOTO o THEN vengono ignorate mentre la clausola ELSE, se presente, viene eseguita. L'esecuzione continua con la successiva istruzione eseguibile. È permesso l'inserimento di una virgola prima di THEN.

Istruzioni IF Nidificate

Le istruzioni IF...THEN...ELSE possono essere nidificate. Tale annidamento è limitato solo dalla lunghezza di una linea. Per esempio:

```
100 IF X>Y THEN PRINT "GREATER" ELSE IF Y<X
 THEN PRINT "LESS THAN" ELSE PRINT "EQUAL"
```

è un'istruzione lecita. Se l'istruzione non contiene lo stesso numero di clausole ELSE e THEN, ogni ELSE viene associato con la clausola THEN successiva non ancora associata a una clausola ELSE. Per esempio:

```
100 IF A=B THEN IF B=C THEN PRINT "A=C"
 ELSE PRINT "A< >C"
```

per A diverso da B non viene visualizzato "A< >B".

Se un'istruzione IF...THEN è seguita da un numero di linea nel modo immediato, si verifica l'errore "Undefined line", a meno che non sia stata precedentemente inserita un'istruzione con il numero di linea specificato.

Osservazioni

Se l'istruzione viene utilizzata per confrontare un dato valore con il risultato di un calcolo tra numeri a virgola mobile, occorre tenere presente che la rappresentazione interna del valore può non essere esatta. Quindi, il confronto deve essere effettuato nell'ambito del grado di accuratezza definito. Per esempio per confrontare la variabile A con il valore 1.0, si utilizza:

```
100 IF ABS (A-1.0)<1.0E-6 THEN...
```

questo confronto risulta vero se il valore di A è uguale ad 1.0 con un errore relativo inferiore a 1.0E-6.

Esempio 1

```
200 IF I THEN GET#1,I
```

Questa istruzione permette di leggere il record di numero I (per I diverso da zero).

Esempio 2

```
100 IF(I<20)*(I>10) THEN DB=1979-1:GOTO 300
110 PRINT "OUT OF RANGE"
 .
 .
 .
```

In questo esempio si controlla se il valore di I è compreso fra 10 e 20. Se I si colloca in questo intervallo, viene calcolato DB e il controllo viene trasferito alla linea 300. In caso contrario, l'esecuzione continua con la linea 110.

Esempio 3

```
210 IF IOFLAG THEN PRINT A$ ELSE LPRINT A$
```

Questa istruzione trasferisce l'output sul terminale o sulla stampante di linea, a seconda del valore della variabile IOFLAG. Se IOFLAG è uguale a zero, l'output è diretto alla stampante di linea, nel caso contrario l'output è diretto al video.

INKEY$ Funzione

Restituisce una stringa di uno o due caratteri, elaborando l'impostazione su tastiera, o una stringa nulla se non vi sono caratteri impostati. INKEY$ viene sempre usata all'interno di un programma.

INKEY$

Caratteristiche

INKEY$ restituisce uno dei seguenti valori:

- Una stringa nulla se non vi sono caratteri in attesa di essere elaborati.
- Una stringa di un carattere in conformità con il carattere impostato da tastiera.
- Una stringa di due caratteri se viene premuto un tasto (o una combinazione di tasti) che non può essere associato ad un codice ASCII standard. Il primo carattere è lo zero esadecimale (00); il secondo indica il codice esteso. Quest'ultimo è di solito, ma non sempre, il codice di scansione del primo tasto premuto (si faccia riferimento al secondo esempio ed a "MS-DOS Guida Utente" per una lista dei codici estesi).

Viene letto un solo carattere anche se vi sono più caratteri in attesa nel buffer della tastiera. Questo valore deve essere quindi assegnato a una variabile prima di essere utilizzato dal programma GW-BASIC.

Osservazioni

Nessun carattere genera un'eco. Tutti i caratteri vengono passati al programma, ad eccezione dei seguenti caratteri di controllo:

PRTSC	stampa il contenuto del video
CTRL NUMLOCK	crea una pausa nel sistema
CTRL BREAK	interrompe il programma
ALT CTRL DEL	riposiziona il sistema

CR viene passato al programma come qualsiasi altro carattere.

Esempio 1

```
100 ' Subroutine di input temporizzato
1010 RESPONSE$ = ""
1020 FOR I% = 1 TO TIMELIMIT%
1030 A$ = INKEY$:IF LEN(A$) = 0 THEN 1060
1040 IF ASC(A$) = 13 THEN TIMEOUT% = 0:RETURN
1050 RESPONSE$ = RESPONSE$ + A$
1060 NEXT I%
1070 TIMEOUT% = 1:RETURN
```

Questo sottoprogramma restituisce due valori:

- RESPONSE$ contiene la stringa impostata da tastiera.
- TIMEOUT% equivale a 0 se l'utente imposta una stringa di caratteri da tastiera prima di completare un dato numero di cicli (TIMELIMIT%), altrimenti equivale a 1.

Esempio 2

Il programma che segue visualizza i risultati di INKEY$.

```
10 S$ = INKEY$
20 IF LEN(S$)=0 THEN GOTO 10
30 IF LEN(S$)=2 THEN GOTO 100
35 'VISUALIZZAZIONE DI CODICI A 1 BYTE
40 PRINT HEX$(ASC(S$))
50 GOTO 10
100 'VISUALIZZAZIONE DI CODICI A 2 BYTE...SECONDO
    BYTE ESADECIMALE (E DECIMALE)
110 S1$=MID$(S$,1,1)): S2$=MID$(S$,2,1)
120 PRINT HEX$(ASC(S1$)); " "; HEX$(ASC(S2$));
    "(";ASC(S2$);")"
130 GOTO 10
```

INP Funzione

Fornisce il byte letto da una porta di input.

INP(*port*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
port	È il un numero di una porta di input, compreso nell'intervallo da 0 a 65535.

Osservazioni

INP è la funzione complementare all'istruzione OUT.

Esempio

```
100 A% = INP (54321)
```

INPUT Istruzione

Permette l'introduzione da tastiera durante l'esecuzione di un programma.

L'istruzione INPUT è usata solo all'interno di un programma.

INPUT [;] [*prompt* ;] *variable*[,*variable*]...

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
prompt	È una costante stringa, racchiusa tra doppi apici, che viene usata per richiedere l'introduzione dei dati da tastiera.
variable	È una variabile numerica o stringa a cui viene assegnato il valore impostato da tastiera.

Caratteristiche

Quando viene incontrata l'istruzione INPUT, l'esecuzione del programma si interrompe e viene visualizzato un punto interrogativo ad indicare che il programma è in attesa di dati. Se è stato definito il parametro *prompt*, la stringa viene visualizzata prima del punto interrogativo, deve quindi essere introdotto il dato richiesto.

Per cancellare il punto interrogativo è possibile usare una virgola, invece di un punto e virgola, dopo la stringa definita in *prompt*.

Se INPUT è seguito da un punto e virgola, **CR** digitato dall'utente per introdurre dati non dà come eco una sequenza di **CR LF**.

Il dato introdotto viene assegnato alle variabili specificate nella lista di variabili. Il numero di dati introdotti deve essere uguale a quello delle variabili della lista. I dati devono essere tra loro separati da virgole.

I nomi di variabili della lista possono essere di tipo numerico o stringa (incluse variabili con indice). Il tipo di dato introdotto deve essere dello stesso tipo specificato dal nome di variabile (non è necessario racchiudere tra doppi apici le stringhe che vengono introdotte in un'istruzione INPUT).

Se a una richiesta di INPUT si risponde con troppi o pochi dati, oppure con un valore non corretto (un valore numerico invece di stringa, ecc.), viene visualizzato il messaggio di errore "?Redo from start"; finché non viene fornita una risposta corretta non viene assegnato alcun valore alle variabili.

In risposta a istruzioni INPUT e LINE INPUT è possibile usare tutte le prestazioni di screen editor GW-BASIC.

Esempio 1

```
10 INPUT X
20 PRINT X "SQUARED IS" X ^ 2
30 END
RUN
? 5
 5 SQUARED IS 25
Ok
```

Esempio 2

```
LIST
10 PI = 3.14
20 INPUT "WHAT IS THE RADIUS";R
30 A = PI*R ^ 2
40 PRINT "THE AREA OF THE CIRCLE IS";A
50 PRINT
60 GOTO 20
Ok
RUN
WHAT IS THE RADIUS? 7.4
THE AREA OF THE CIRLCE IS 171.9464

WHAT IS THE RADIUS?
```

INPUT # Istruzione

Legge dati da un file sequenziale su disco e li assegna a variabili di programma.

L'istruzione INPUT# viene di solito usata all'interno di un programma.

INPUT #*filenum*, *variable* [,*variable*]...

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filenum	È il numero usato quando il file è stato aperto per le operazioni di input.
variable	È una variabile numerica o stringa che riceve un dato dal file. (Il tipo di dati del file deve essere compatibile con il tipo specificato dal nome di variabile.) Nessun punto interrogativo viene visualizzato con INPUT#.

Caratteristiche

I dati del file devono essere dello stesso tipo di quelli digitati in risposta ad un'istruzione INPUT. Con i valori numerici vengono ignorati spazi iniziali, caratteri di ritorno a capo e di avanzamento linea; quando viene incontrato un carattere diverso da questi GW-BASIC lo considera come primo carattere di un numero. Il numero termina quando viene incontrato uno spazio, un ritorno a capo, un avanzamento linea o una virgola.

Quando GW-BASIC scandisce il file sequenziale per ricercare una stringa, ignora eventuali spazi iniziali, caratteri di ritorno a capo e di avanzamento linea; quando viene incontrato un carattere diverso da questi GW-BASIC lo considera come primo carattere stringa. Se questo primo carattere è doppio apice ("), la stringa comprende i tutti i caratteri letti, fino al successivo carattere doppio apice; perciò una stringa inclusa tra doppi apici non può comprendere il carattere doppio apice. Se il primo carattere della stringa non è un carattere doppio apice, la stringa non è compresa tra apici e terminerà con una virgola, un ritorno a capo o un

avanzamento linea (o dopo che sono stati letti 255 caratteri). Se, mentre viene introdotto un valore numerico o stringa, viene raggiunta la fine del file il campo termina.

Esempio

```
100 INPUT #1,X$,Y$,Z$
```

In questo esempio viene usata l'istruzione INPUT # per leggere dati da un file sequenziale e metterli a diposizione del programma.

INPUT$ Funzione

Restituisce una stringa di caratteri letti dal device di input standard, dalla tastiera o da un file.

La funzione INPUT$ viene usata all'interno di un programma e ne sospende l'esecuzione finché non vengono introdotti tutti i caratteri richiesti.

INPUT$(*length*[,[#]*filenum*])

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
length	È un'espressione intera che specifica il numero di caratteri che devono essere impostati su tastiera o letti da un file.

ELEMENTO DI SINTASSI	SIGNIFICATO
filenum	È il numero di file che specifica il file da leggere. Se viene omesso *filenum*, vengono letti i dati impostati sulla tastiera.

Caratteristiche

Se i dati vengono presi da tastiera, non viene visualizzato nessun carattere. Tutti i caratteri (compresi i caratteri di controllo) vengono inoltrati al programma, ad eccezione di **CTRL BREAK**, che viene usato per interrompere l'esecuzione della funzione INPUT$.

Per i file di comunicazione viene preferita la funzione INPUT$ alle istruzioni INPUT# e LINE INPUT#, perché in questo caso tutti i caratteri ASCII possono essere significativi. È meno consigliabile usare INPUT#, poiché l'input termina non appena vengono introdotti virgola o **CR** oppure LINE INPUT#; in questo caso l'input termina quando viene incontrato **CR**.

Esempio 1

```
5  'LISTA I CONTENUTI DI UN FILE SEQUENZIALE IN
    ESADECIMALE
10 OPEN "I",1,"DATA"
20 IF EOF(1) THEN 50
30 PRINT HEX$(ASC(INPUT$(1, #1)));
40 GOTO 20
50 PRINT
60 END
```

Esempio 2

```
 .
 .
 .
100 PRINT "TYPE P TO PROCEED OR S TO STOP"
110 X$=INPUT$(1)
120 IF X$="P" THEN 500
130 IF X$="S" THEN 700 ELSE 100
 .
 .
 .
```

Esempio 3

Questa sequenza di istruzioni può essere usata per leggere un file di comunicazione.

```
10 WHILE NOT EOF(1)
20   A$=INPUT$(LOC(1),#1)
30 ...
40 ... Elabora i dati contenuti in A$...
50 ...
60 WEND
```

La sequenza di istruzioni precedente può essere così interpretata: quando la coda di input non è vuota, restituisce il numero di caratteri in essa contenuti e li memorizza in A$; se ci sono più di 255 caratteri, ne restituisce solo 255 per volta, onde evitare l'errore "String overflow". Inoltre, in questo caso la condizione EOF(1) è falsa e l'input continua finché la coda di input è vuota.

INSTR Funzione

Ricerca la prima occorrenza di una data sottostringa all'interno di una stringa e fornisce la posizione in cui essa viene individuata.

INSTR([*start*,] *string*,*substring*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
start	È un'espressione intera compresa tra 1 e 255, che specifica il punto da cui iniziare la ricerca. Se *start* viene omesso si assume il valore 1.
string	È un'espressione stringa (in particolare una costante o una variabile stringa) che indica la stringa in cui cercare.
substring	È un'espressione stringa (in particolare una constante o una variabile stringa) di cui si deve cercare la prima occorrenza.

Caratteristiche

SE...	ALLORA...
start >LEN	il valore restituito è 0.
start <1 oppure *start* >255	si verifica l'errore "Illegal function call".
string è una stringa nulla	il valore restituito è 0.
substring non viene trovato	il valore restituito è 0.
substring è una stringa nulla e *start* viene specificato	il valore restituito è uguale a quello di *start*.
substring è una stringa nulla e *start* non viene specificato	il valore restituito è 1.

Esempio

```
10 X$="ABCDEB"
20 Y$="B"
30 PRINT INSTR(X$,Y$); INSTR(4,X$,Y$)
RUN
 2  6
Ok
```

La posizione in cui viene trovata la prima occorrenza della sotto-stringa viene sempre calcolata dall'inizio della stringa originaria, anche se viene specificato *start*.

INT Funzione

Calcola l'intero più grande minore o uguale all'argomento.

INT(*numexp***)**

Caratteristiche

Si faccia riferimento alle funzioni CINT e FIX, che pure restituiscono valori interi.

Esempi

```
PRINT INT(99.89)
 99
Ok
```

```
PRINT INT(-12.11)
-13
Ok
```

IOCTL Istruzione

Invia una stringa di "Dati di Controllo" ad un Character Device Driver ogni volta che il Driver è stato aperto (OPEN). Generalmente l'istruzione IOCTL viene usata all'interno di un programma.

IOCTL [#]*filenum*,*string*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filenum	È il numero di file aperto al Driver di Device.
string	È un'espressione stringa che contiene i Dati di Controllo.

Caratteristiche

I comandi di IOCTL in genere consistono in 2 o 3 caratteri opzionali seguiti da un argomento alfanumerico. Una stringa IOCTL può raggiungere la lunghezza di 255 byte.

L'istruzione IOCTL funziona solo se:

- il device driver è installato
- il device driver elabora le stringhe di IOCTL
- l'interprete GW-BASIC esegue un'istruzione OPEN su un file di quel device.

La maggior parte dei device driver di MS-DOS standard non elabora le stringhe IOCTL, perciò per il programmatore è necessario determinare se il driver specificato è in grado di gestire il comando.

Se l'utente ha installato il proprio driver per sostituire LPT1 e quel driver è in grado di determinare la Lunghezza di Pagina, allora un comando IOCTL per determinare o cambiare la Lunghezza di Pagina potrebbe essere:

PL*n*

Dove: *n* è la nuova Lunghezza di Pagina.

Esempio 1

Per aprire il nuovo driver LPT1 e fissare la Lunghezza di Pagina pari a 66 linee possono essere usate le seguenti istruzioni:

```
10 OPEN "LPT1:" FOR OUTPUT AS #1
20 IOCTL #1, "PL66"
```

Esempio 2

```
10 OPEN "\DEV\LPT1" FOR OUTPUT AS #1
20 IOCTL #1, "PL56"
```

Anche questa istruzione serve per aprire LPT1, ma con una Lunghezza iniziale di Pagina pari a 56 linee.

Sono possibili altri comandi di IOCTL definiti dall'utente come PT*n* (fissa la Tabulazione di Stampa ogni *n* spazi).

Errori Possibili

"Bad file number": l'istruzione IOCTL è diretta a un driver che non è stato aperto.

"Illegal function call": il device non elabora la stringa IOCTL.

"Device fault": è presente un errore nei dati di controllo.

IOCTL$ Funzione

Trasferisce una stringa di "Dati di Controllo" da un Character Device Driver che risulta aperto. IOCTL$ viene generalmente usata all'interno di un programma.

IOCTL$([#]*filenum*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filenum	È il numero di file aperto al device.

Caratteristiche

La funzione IOCTL$ è usata più frequentemente per ricevere conferma della corretta esecuzione di una istruzione IOCTL o per ottenere informazioni sullo stato corrente.

IOCTL$ può essere usata per chiedere ad un device di comunicazione di riportare la velocità di trasmissione corrente (baud-rate), le informazioni sull'ultimo errore, l'ampiezza della linea logica, ecc.

La funzione IOCTL$ funziona solo se:

- il device driver è installato
- il device driver elabora le stringhe IOCTL
- l'interprete GW-BASIC esegue un'istruzione OPEN su un file di quel device.

Esempio

```
10 OPEN "\DEV\FN1" AS #1
20 IOCTL #1,"RAW" 'Informa il device che i dati sono "RAW"
30 IF IOCTL$(1)="0" THEN CLOSE 1
```

Se il Character Driver FN1 restituisce un valore falso alla richiesta IOCTL relativa ai dati RAW, allora è necessario chiudere il file e interrompere l'elaborazione.

Errori Possibili

"Bad file number": l'istruzione IOCTL è diretta a un driver che non è stato aperto.

"Illegal function call": il device non elabora la stringa IOCTL.

KEY Istruzione

Permette di utilizzare un tasto funzione per introdurre automaticamente una sequenza di caratteri. Altre opzioni permettono di abilitare o disabilitare la visualizzazione del tasto funzione sulla 25 riga o listare i valori del tasto funzione.

KEY { **OFF** I **ON** I **LIST** I*n,stringexp* }

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È il key number cioé un'espressione che restituisce un intero senza segno compreso tra 1 e 10.
stringexp	È un'espressione stringa assegnata al tasto. Le costanti stringa devono essere racchiuse tra doppi apici. Il parametro *stringexp* può essere lungo fino a un massimo di 15 caratteri. Stringhe più lunghe vengono troncate a 15 caratteri.

Caratteristiche

L'istruzione KEY permette la definizione di un tasto funzione come Soft Key cioé è possibile generare qualsiasi sequenza di caratteri premendo tale tasto.

ISTRUZIONE	SIGNIFICATO
KEY OFF	Cancella la visualizzazione Soft Key sull'ultima riga del video, rendendo quest'ultima disponibile per il programma utente GW-BASIC. In questo caso è possibile usare LOCATE 25, 1 seguita da PRINT per visualizzare dati sull'ultima riga del video. KEY OFF non disabilita i tasti funzione.
KEY ON	Visualizza sull'ultima riga i valori Soft Key. Se l'ampiezza del video è 80, vengono visualizzati dieci Soft Key. Se l'ampiezza del video è 40, ne vengono visualizzati solo cinque. In entrambe le ampiezze del video, vengono visualizzati solo i primi 6 caratteri di ogni valore. Se non vengono visualizzati tutti i tasti funzione, è possibile far scorrere la visualizzazione dei tasti funzione (incrementando, ogni volta, di uno il numero del tasto più a sinistra) premendo **CTRL T**. ON è lo stato di default.
KEY LIST	Visualizza tutti i 10 valori Soft Key. Vengono visualizzati tutti i 15 caratteri di ogni valore.
KEY *n*, *stringexp*	Specifica il tasto funzione *n*. Ognuno dei 10 Tasti Funzione può essere associato ad una stringa al massimo di 15 byte tramite KEY *n,stringexp*. Quando viene premuto il tasto, la stringa associata diventa l'input per GW-BASIC.

ISTRUZIONE	SIGNIFICATO
	Inizialmente, i tasti Soft Key assumono i seguenti valori di default: F1 - LIST space F2 - RUN CR F3 - LOAD" F4 - SAVE" F5 - CONT CR F6 - ,"LPT1:" CR F7 - TRON CR F8 - TROFF CR F9 - KEY space F10 - SCREEN 0,0,0 CR

Osservazioni

1. Se il valore dato per il tasto funzione non si colloca nell'intervallo da 1 a 10, si verifica l'errore "Illegal function call". Viene in questo caso mantenuta la precedente associazione tasto/stringa.

2. La stringa associata al tasto può essere lunga da 1 a 15 caratteri. Se la stringa contiene più di 15 caratteri, vengono presi solo i primi 15.

3. L'associazione di una stringa nulla (una stringa di lunghezza uguale a 0) ad un tasto Soft Key inibisce l'uso del tasto funzione come Soft Key.

4. Quando ad un tasto funzione viene associata la stringa Soft Key, la funzione INKEY$ restituisce un carattere alla volta della stringa Soft Key.

5. Se viene disabilitata la funzione Soft Key (vedere il punto 3), INKEY$ restituisce una stringa di due caratteri: il primo è uno zero esadecimale (00), il secondo è lo "Scan-code" del tasto.

Esempi

```
50 KEY ON
```

Visualizza i tasti Soft Key sull'ultima linea.

```
600 KEY OFF
```

Cancella la visualizzazione Soft Key.

```
70 KEY 1,"MENU" + CHR$(13)
```

Assegna la stringa "MENU" **CR** al tasto Soft Key 1. Tali assegnazioni possono venire usate per una rapida introduzione di dati.

```
80 KEY 2,""
```

Disabilita la funzionalità Soft Key del tasto 2.

La routine seguente inizializza i primi 5 tasti Soft Key:

```
1 KEY OFF 'Visualizza i tasti spenti
10 DATA KEY1, KEY2, KEY3, KEY4, KEY5
20 FOR I=1 TO 5: READ SOFTKEYS$(I)
30 KEY I, SOFTKEYS$(I)
40 NEXT I
50 KEY ON 'Visualizza i nuovi Soft Key
```

Definizione dei Tasti 15-20

La definizione dei tasti da 15 a 20 permette di rilevare eventuali Tasti-Ctrl, Tasti-Shift o Tasti Super-Shift (**ALT**). Questi tasti vengono definiti dall'istruzione:

KEY *n* **, CHR$(***shift***) + CHR$(***scan-code***)**

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È un'espressione intera compresa nell'intervallo da 15 a 20.
shift	È un valore numerico corrispondente ai seguenti valori esadecimali: **CAPS LOCK** &H40 (è attivo Caps Lock) **NUM LOCK** &H20 (è attivo Num Lock, solo sulla Tastiera 1) **ALT** &H08 (viene premuto il tasto Alt) **CTRL** &H04 (viene premuto il tasto CTRL) **SHIFT** destro &H01 **SHIFT** sinistro &H02 Possono essere usati entrambi i tasti **SHIFT** (sia sinistro che destro), dove i valori di &H01, &H02 o &H03 (la somma di 01 esadecimale e 02 esadecimale) denotano un tasto **SHIFT**. È anche possibile aggiungere shift multipli, come i tasti **CTRL** e **ALT** insieme, aggiungendo i valori dello shift associato.
scan-code	È un numero decimale compreso tra 1 e 83 sulla Tastiera 1 e tra 1 e 103 sulla Tastiera 2. Esso rappresenta lo "scan-code" (decimale) del tasto da rilevare.

Osservazioni

Se vengono premuti diversi tasti funzione, mentre sono temporaneamente disattivati (da un'istruzione KEY(*n*) STOP esplicita o implicita), vengono richiamate le corrispondenti routine di rilevamento nell'ordine in cui i tasti vengono riattivati. Se vengono riattivati simultaneamente diversi tasti rilevati essi vengono elaborati nell'ordine seguente:

1. **CTRL PRT SC**, anche se definito come tasto rilevabile, continua a produrre una copia stampata di tutto il testo inviato su video.

2. I tasti funzione da **F1** a **F10** e il cursore danno una direzione ai tasti. L'associazione di un tasto funzione o di un tasto di movimento cursore con il rilevamento del tasto definito dall'utente non provoca nessun effetto, poiché questi tasti sono considerati pre-definiti.

3. Vengono esaminati i tasti definiti dall'utente (15-20).

Tutti i tasti rilevati non vengono passati a GW-BASIC, cioé non vengono passati al buffer della tastiera. Questo vale per tutti i tasti, compreso **CTRL BREAK** o **CTRL ALT DEL**. In questo modo è possibile evitare l'interruzione accidentale di un programma o la riattivazione del sistema.

Esempi

Vedere l'istruzione ON KEY(*n*) GOSUB.

KEY(*n*) Istruzione

KEY(*n*) ON abilita, KEY(*n*) OFF disabilita e KEY(*n*) STOP sospende il rilevamento di eventi del tasto specificato.

L'istruzione KEY(*n*) viene generalmente usata all'interno di un programma.

KEY(*n*) { ON | OFF | STOP}

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È un'espressione intera, il cui valore è compreso tra 1 e 20, che specifica il numero di un tasto funzione (se è compreso tra 1 e 10), un tasto di direzione del cursore (se è compreso tra 11 e 14) o un tasto definito dall'utente (se è compreso tra 15 e 20). I tasti di direzione del cursore sono: Cursor up (11) Cursor left (12) Cursor right (13) Cursor down (14). I tasti definiti dall'utente sono specificati dalla formula: KEY*n*, CHR$(*shift*) + CHR$(*scan-code*) riportata per l'istruzione KEY.

Per Abilitare o Disabilitare il Rilevamento di un Tasto *n*

SE...	ALLORA...
viene eseguita l'istruzione KEY(*n*) ON.	viene abilitato il rilevamento del tasto *n*.
viene eseguita l'istruzione KEY(*n*) OFF	viene disabilitato il rilevamento del tasto *n*.
viene eseguita l'istruzione KEY(*n*) STOP	viene sospeso il rilevamento del tasto *n*. Premendo KEY(*n*) viene memorizzato l'evento; in questo modo, quando viene eseguita un'istruzione KEY(*n*) ON, viene attivata la routine di rilevamento.
viene eseguita l'istruzione ON KEY(*n*) GOSUB	viene sospeso il rilevamento del tasto *n*.
ha luogo un rilevamento di errore	viene automaticamente disabilitato tutto il rilevamento.

Esempio

```
10 KEY 11, "SCREEN 0,0,0" 'assegna il tasto 11
20 KEY (11) ON 'abilita il rilevamento del tasto
  :
100 ON KEY (11) GOSUB 1000
  :
Premuto il tasto 11
  :
1000 REM Routine di rilevamento di KEY (11)
```

KILL Comando

Cancella un file su disco.

KILL { *filespec* | *pathname* }

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filespec (o *pathname*)	È un'espressione stringa che specifica il file da cancellare. Il nome di file deve comprendere anche l'eventuale estensione.

Caratteristiche

KILL controlla se il file è aperto e in questo caso visualizza l'errore "File already open". KILL, come OPEN, non può riconoscere un file in una directory diversa da quella che è stata aperta. È quindi possibile, anche in questi casi, che si verifichi l'errore "File already open".

Il comando KILL può essere usato solo per cancellare un file. Per cancellare una directory è necessario usare il comando RMDIR.

Esempio

```
200 KILL "A:DATA1.DAT"
300 KILL "C:DIR1\DIR2\PROG2.BAS"
```

Il nome di file deve comprendere l'eventuale estensione. GW-BASIC non fornisce l'estensione .BAS per il comando KILL.

LCOPY Comando

Stampa il contenuto del video (testo e grafica). Per stampare un contenuto grafico occorre avere una stampante grafica e aver prima eseguito, in MS-DOS, il comando GRAPHICS.

LCOPY [*n*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È un argomento fittizio e può essere fornito qualsiasi valore. Questo parametro è permesso solo per compatibilità con altri linguaggi BASIC dove specifica che deve essere copiato un testo o un'immagine grafica.

LEFT$ Funzione

Fornisce una sottostringa estraendo i caratteri più a sinistra di una stringa data, per una lunghezza pari a quella specificata dal parametro *length*.

LEFT$(*string*,*length*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
string	È un'espressione stringa il cui valore corrisponde alla stringa dalla quale la sottostringa deve essere estratta.
length	È un'espressione intera, compresa tra 0 e 255, che specifica il numero di caratteri che devono essere restituiti.

Caratteristiche

Se *length* è maggiore di LEN (string), viene restituita l'intera stringa iniziale. Se *length* = 0, viene restituita una stringa nulla (di lunghezza zero).

Si vedano anche le funzioni MID$ e RIGHT$.

Esempio

```
Ok
10 A$ = "GW-BASIC"
20 B$ = LEFT$(A$,6)
30 PRINT B$
RUN
GW-BAS
Ok
```

LEN Funzione

Calcola la lunghezza di una data stringa.

LEN(*stringexp*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
stringexp	È una qualsiasi espressione stringa, di cui si desidera calcolare la lunghezza.

Caratteristiche

Nel numero dei caratteri calcolato vengono inclusi anche caratteri non stampabili e spazi vuoti.

Se l'argomento *stringexp* è una stringa nulla, LEN restituisce zero.

Esempio

```
10 X$ = "PORTLAND, OREGON"
20 PRINT LEN(X$)
RUN
 16
Ok
```

LET Istruzione

Assegna un valore ad una variabile.

[**LET**] *variable* = *expression*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
variable	È una variabile numerica o stringa a cui viene assegnato il valore dell'espressione.
expression	È un'espressione il cui valore viene assegnato alla variable che si trova alla sinistra del segno uguale.

Caratteristiche

Il tipo dell'espressione (numerica o stringa) deve essere uguale al tipo della variabile; nel caso contrario si verifica l'errore "Type mismatch". Comunque, nelle assegnazioni numeriche il tipo dell'espressione (intero, semplice o doppia precisione) può essere diverso dal tipo della variabile di destinazione. In questo caso GW-BASIC converte il valore dell'espressione nel tipo della variabile. La conversione può dare luogo a un arrotondamento oppure a un overflow.

La parola LET è opzionale. Quando viene assegnata un'espressione a un nome di variabile, è sufficiente il segno uguale.

Esempio

```
110 LET D=12
120 LET E=12^2
130 LET F=12^4
140 LET SUM=D+E+F
150 A(I)=300
160 A$(K)="ABC"
 .
 .
 .
```

LINE Istruzione

Traccia una linea, un rettangolo o un rettangolo colorato (solo modo grafica).

LINE [[**STEP**] (*x1, y1*)] **-** [**STEP**] (*x2,y2*) [,[*color*] [,**B**[**F**]][,*style*]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
(*x1*,*y1*), (*x2*,*y2*)	Rappresentano coordinate assolute o relative. Se (*x1*,*y1*) viene omesso, viene assunto l'ultimo punto a cui si è fatto riferimento.
color	È il numero di colore (compreso nell'intervallo da 0 a 3) che specifica il colore con il quale deve essere tracciata la linea (o il rettangolo). In Media Risoluzione, *color* sceglie il colore dalla tavolozza attiva; in Alta e Altissima Risoluzione, un valore di *color* pari a 0 o 2 indica il nero e un valore pari a 1 o 0 indica il bianco. Se *color* è omesso, il colore è dato dal valore del parametro *gforeground* dell'istruzione COLOR attiva; se pure questo parametro è omesso viene assunto 3 in Media Risoluzione e 1 in Alta e Altissima Risoluzione.
B	Indica che deve essere tracciato un rettangolo con il colore selezionato da *color*.
F	Indica che l'area del rettangolo deve essere colorata.
style	Parametro di 16 bit, di norma espresso in formato esadecimale, che specifica lo "stile" della linea. Non ha alcun effetto se è stato specificato BF.

Caratteristiche

L'esempio che segue traccia una linea dall'ultimo punto a cui si è fatto riferimento al punto specificato (x2,y2). Poiché non viene specificato nessun colore, viene preso il colore di primo piano di default per le figure.

LINE -(x2,y2)

Gli esempi seguenti specificano i punti iniziali e finali, espressi in coordinate assolute.

LINE (10,10)-(319,199)	'traccia sul video una linea diagonale
LINE (10,100)-(319,100)	'traccia sul video una linea orizzontale

È possibile specificare il numero di colore della linea:

LINE (15,15)-(25,25),2	'traccia una linea con il numero di colore 2.

Il parametro B viene usato per tracciare un rettangolo nel colore di primo piano di default, dove i punti (*x1,y1*) e (*x2,y2*) rappresentano gli angoli opposti. Nell'esempio seguente non viene specificato nessun numero di colore:

LINE (10,10)-(100,100),,B	'traccia un rettangolo nel colore di primo piano di default

per includere il colore si può procedere come segue:

LINE (10,10)-(200,200),2,BF	'rettangolo colorato con il colore 2

L'opzione B consente una programmazione più semplice per tracciare un rettangolo che altrimenti richiederebbe l'uso di quattro istruzioni:

LINE *(x1,y1)-(x2,y1)*
LINE *(x1,y1)-(x1,y2)*
LINE *(x2,y1)-(x2,y2)*
LINE *(x1,y2)-(x2,y2)*

L'opzione BF colora l'area del rettangolo con il colore selezionato da *color.*

Se la seconda coppia di coordinate viene preceduta dall'opzione STEP, queste coordinate sono interpretate come relative al primo punto specificato. Per esempio:

LINE (50,50) -STEP(15,-13)

traccia una linea dal punto (50,50) al punto (65,37).

Stile della Linea

Nell'istruzione LINE può anche essere definito l'elemento opzionale *style*, che è un valore intero su 16 bit, atto a specificare una maschera per lo stile della linea. *style* viene usato per tracciare linee e rettangoli ma non per colorare l'area di un rettangolo.

Ogni volta che LINE illumina un pixel su video, viene utilizzato il valore corrente del bit *style*. Se il bit è uguale a 0 non viene modificato il colore del pixel, se il bit è uguale a 1 viene illuminato un pixel nel colore specificato. Per ogni pixel successivo viene selezionata la posizione di bit successiva in *style*. Quando l'intera maschera è stata scandita la scansione riprende dall'inizio fino al completamento della linea.

Poiché un bit uguale a 0 in *style* non modifica il colore del pixel, è possibile sovrapporre una linea "styled" su una linea precedente di un dato colore per avere un colore di fondo valido solo per quella linea.

Per esempio:

LINE (0,0)-(160,100),3,,&HFF00

traccia una linea tratteggiata dal vertice in alto a sinistra al centro del video, assumendo che il video abbia un'ampiezza di 320 pixel e un'altezza di 200.

LINE INPUT Istruzione

Permette l'introduzione di un'intera linea (fino a 254 caratteri) e la assegna a una variabile stringa, senza far uso di delimitatori. L'istruzione LINE INPUT viene usata solo all'interno di un programma.

LINE INPUT[;] [*prompt*;]*stringvar*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
prompt	È una costante stringa racchiusa tra doppi apici che viene visualizzata prima dell'introduzione dei dati. Non viene visualizzato nessun punto interrogativo, a meno che non faccia parte di *prompt*.
stringvar	È il numero di una variabile stringa a cui viene assegnata la linea.

Caratteristiche

Tutti i caratteri introdotti dalla fine di *prompt* a **CR** vengono assegnati a *stringvar*.

Eventuali spazi in coda vengono ignorati. Se si incontra **LF CR** ne viene generato l'eco, ma **CR** viene ignorato, **LF** viene inserito in *stringvar* e l'input dei dati continua.

Se l'istruzione LINE INPUT è immediatamente seguita da un punto e virgola viene eliminato l'eco di **CR**, cioé la sequenza **CR LF**.

Un'istruzione LINE INPUT può essere interrotta tramite **CTRL BREAK**. GW-BASIC ritorna quindi in stato comandi e visualizza "Ok"; l'esecuzione viene ripresa da LINE INPUT tramite il comando CONT.

È possibile usare tutte le prestazioni di screen editor di GW-BASIC con le istruzioni INPUT e LINE INPUT.

Esempio

Vedere l'istruzione LINE INPUT,#.

LINE INPUT# Istruzione

Legge un'intera linea (fino a 254 caratteri), esclusi i delimitatori, da un file dati sequenziale su disco e la assegna ad una variabile stringa.

L'istruzione LINE INPUT# viene di solito usata dall'interno di un programma.

LINE INPUT# *filenum,stringvar*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filenum	È il numero con cui il file è stato aperto.
stringvar	È la variabile stringa a cui viene assegnata la linea.

Caratteristiche

L'istruzione LINE INPUT# legge tutti i caratteri del file sequenziale fino a **CR**, quindi salta la sequenza **CR LF**. La successiva istruzione LINE INPUT# legge tutti i caratteri fino al successivo **CR**. (Se viene incontrata una sequenza di **LF CR**, viene conservata.)

LINE INPUT# è particolarmente utile se ogni linea di un file dati è stata suddivisa in campi, oppure se un programma GW-BASIC in formato ASCII viene letto da un altro programma come un file dati (vedere il comando SAVE).

Esempio

```
10 OPEN "O",1,"LIST"
20 LINE INPUT "CUSTOMER INFORMATION?";C$
30 PRINT #1, C$
40 CLOSE 1
50 OPEN "I",1,"LIST"
60 LINE INPUT #1, C$
70 PRINT C$
80 CLOSE 1
RUN
CUSTOMER INFORMATION? LINDA JONES 234,4 MEMPHIS
LINDA JONES   234,4   MEMPHIS
Ok
```

LIST Comando

Lista su video, su un file o su un device specificato il programma corrente.

Il comando LIST viene di solito usato in modo immediato.

LIST [*linenum1*][-[*linenum2*]][, { *device* | *filespec* | *pathname* }]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
linenum1 e *linenum2*	Sono i numeri della prima e dell'ultima linea da listare. Per indicare la linea corrente è possibile usare un punto (.) in luogo di uno dei due numeri di linea.
device	È un'espressione stringa utilizzata per specificare un device.
filespec (o *pathname*)	È un'espressione stringa utilizzata per specificare un file.

Caratteristiche

Se vengono omessi i parametri opzionali *device*, *filespec* o *pathname* le linee specificate vengono listate su video.

È possibile interrompere in qualsiasi momento il listato sul video o sulla stampante premendo **CTRL BREAK**; non è possibile interromperlo su un file o su un device: in questo caso il listato continua fino alla fine.

Se viene omesso l'intervallo delle linee, viene listato l'intero programma.

La sintassi permette le seguenti opzioni:

- Se viene specificato solo *linenum1*, vengono listate quella linea e tutte le linee successive.

- Se viene specificato solo *linenum2*, vengono listate tutte le linee dall'inizio del programma fino alla linea data.

- Se vengono specificati entrambi i numeri, vengono listate tutte le linee appartenenti all'intervallo specificato.

Quando il listato è diretto a un file su disco, la parte specificata del programma viene registrata in formato ASCII. In seguito il file può essere trattato con il comando MERGE.

Esempi

LIST ,LPT1:

Lista il programma sulla stampante di linea.

LIST 10-90

Lista su video le linee da 10 a 90.

LIST 10-,SCRN:

Lista su video le linee da 10 alla fine del programma.

LLIST Comando

Lista sulla stampante il programma corrente. Il comando LLIST viene di solito usato in modo immediato.

LLIST [*linenum1*][-[*linenum2*]]

Caratteristiche

Per il significato dei parametri occorre fare riferimento al comando LIST. LLIST assume una riga di stampa pari a 132 caratteri.

Dopo l'esecuzione di un comando LLIST, GW-BASIC ritorna sempre in stato comandi.

Esempi

LLIST

Lista un programma completo.

LLIST 50

Lista la linea 50.

LLIST 20-40

Lista le linee 20-40.

LLIST-150

Lista a partire dalla prima linea del programma fino alla linea 150.

LLIST 90-

Lista tutte le linee del programma a partire da 90 fino alla fine del programma.

LOAD Comando

Carica in memoria un programma da un drive specificato e, se viene specificata l'opzione R, lo esegue.

LOAD { *filespec* I *pathname* } [**,R**]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filespec (o *pathname*)	È un'espressione stringa che specifica il file da caricare e, opzionalmente, un drive. Se *filespec* viene omesso, viene assunto il drive di default di MS-DOS.
R	Rappresenta Run. È opzionale e permette che il programma caricato venga anche eseguito a partire dalla prima istruzione. In questo caso tutti i file dati aperti vengono mantenuti aperti.

Caratteristiche

Prima di caricare il programma specificato, LOAD cancella tutte le variabili e le linee di programma in quel momento residenti in memoria e chiude tutti i file dati aperti. Se però viene specificata l'opzione R tutti i file dati aperti vengono mantenuti aperti e il programma viene caricato ed eseguito.

Si noti che:

RUN *filespec* è equivalente a LOAD *filespec*, R.

e che:

RUN *pathname* è equivalente a LOAD *pathname*, R.

Esempi

LOAD "STRTRK",R

Carica ed esegue il programma STRTRK.BAS

LOAD "B:MYPROG"

Carica il programma MYPROG.BAS dal disco sul drive B, ma non lo esegue.

LOC Funzione

Restituisce la posizione corrente sul file. La funzione LOC viene di solito usata all'interno di un programma.

LOC(*filenum*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filenum	È il numero con cui il file è stato aperto.

Caratteristiche

Per file su disco ad accesso diretto, la funzione LOC fornisce il numero del record appena letto, tramite l'istruzione GET, o scritto, tramite l'istruzione PUT.

Per file sequenziali, la funzione LOC restituisce la posizione corrente, in byte, sul file divisa per 128.

Quando un file è aperto per operazioni di APPEND o OUTPUT, LOC restituisce la dimensione del file in byte/128.

Per file di comunicazione, la funzione LOC restituisce il numero di caratteri in attesa di essere letti che sono presenti nella coda di input. Se la coda di input contiene più di 255 caratteri, LOC restituisce 255. Poiché una stringa è limitata a 255 caratteri, il programmatore non deve controllare la lunghezza della stringa prima di leggervi dei dati. Se nel buffer sono contenuti meno di 255 caratteri i valori restituiti da LOC dipendono dal modo (ASCII o binario) in cui il device è stato aperto.

In entrambi i modi, LOC restituisce il numero di caratteri che possono essere letti dal device. Comunque, in modo ASCII, le routine di basso livello interrompono l'accodamento dei caratteri non appena raggiunta la condizione di end-of-file. Il carattere che identifica end-of-file non viene accodato e non può venire letto. Se viene fatto il tentativo di leggerlo si verifica l'errore "Input past end".

Esempio

```
100 IF LOC(2)>100 THEN STOP
```

LOCATE (Testo) Istruzione

Sposta il cursore nella posizione specificata sulla pagina attiva. LOCATE può anche visualizzare o rendere invisibile il cursore utente e definire la sua dimensione o la dimensione dei cursori utente e di scrittura.

LOCATE [*row*][,[*column*][,[*cursor*][,[*start*][,*stop*]]]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
row	È il numero di linea del video; è un'espressione numerica che restituisce un intero senza segno compreso tra 1 e 25.
column	È il numero di colonna del video; è un'espressione numerica che restituisce un intero senza segno compreso tra 1 e 80, a seconda dell'ampiezza del video.
cursor	È un valore booleano che indica quando il cursore utente è visibile o no. Il valore zero rende invisibile il cursore utente, un valore diverso da zero (per esempio 1) lo visualizza.

ELEMENTO DI SINTASSI	SIGNIFICATO
start	È un'espressione numerica il cui valore intero rappresenta la scanline iniziale del cursore. Se *start* è compreso tra 0 e 7 viene definita la dimensione di entrambi i cursori (utente e di scrittura). Se *start* è compreso tra 32 e 39, viene definita solo la dimensione del cursore utente. Altri valori non sono consentiti.
stop	È un'espressione numerica il cui valore intero rappresenta la scanline finale del cursore, ha lo stesso intervallo di valori di *start* e deve essere maggiore o uguale a *start*; se è uguale il cursore si riduce ad una sola scanline. Se questo parametro è omesso e viene specificato *start*, *stop* assume il valore di *start*.

Caratteristiche

In GW-BASIC ci sono tre cursori:

- Il cursore inserimento, che compare quando l'utente inserisce caratteri sulla linea GW-BASIC corrente.
- Il cursore di scrittura, che compare durante l'introduzione di una linea GW-BASIC e durante l'introduzione di dati, in risposta a istruzioni INPUT.
- Il cursore utente che può essere visualizzato durante l'esecuzione del programma, quando non si sta eseguendo l'istruzione INPUT. In particolare può essere utile visualizzare questo cursore durante l'esecuzione delle funzioni INPUT$ e INKEY$, quando il cursore di scrittura non compare.

Il cursore di scrittura è il piú usato, viene inizializzato come una sottolineatura di 2 scanline di spessore; il cursore inserimento è alto 4 scanline; il cursore utente è inizialmente disabilitato (ma la sua dimensione viene inizializzata a quella della matrice dei caratteri). Il cursore inserimento ha una dimensione fissa; le dimensioni del cursore di scrittura e utente possono invece essere programmate.

Con l'istruzione LOCATE è possibile omettere qualsiasi parametro, questi assumono il valore corrente.

Visualizzazione di Caratteri sulla Linea 25

Generalmente in GW-BASIC, la linea 25 del video non viene utilizzata poiché è riservata alla visualizzazione dei tasti funzione.

Essa può venire disattivata, usando l'istruzione KEY OFF; si usi poi LOCATE 25,1: PRINT... per visualizzare i caratteri sulla linea 25. Le istruzioni PRINT sulla linea 25 devono terminare con un punto e virgola, altrimenti in determinate circostanze la videata può scorrere. Anche le istruzioni seguenti consentono la visualizzazione del testo su 25 linee con scorrimento, senza una linea dedicata ai tasti funzione:

KEY OFF
VIEW PRINT

Esempi

100 LOCATE 1,1

Sposta il cursore nella posizione iniziale nell'angolo in alto a sinistra.

200 LOCATE ,,1

Visualizza il cursore utente; la sua posizione rimane immutata.

```
300 LOCATE ,,0
```

Rende invisibile il cursore utente.

```
400 LOCATE 6,1,1,0,7
```

Sposta il cursore alla linea 6 colonna 1. Visualizza il cursore utente coprendo l'intera matrice dei caratteri, iniziando dalla scanline 0 e terminando alla scanline 7.

LOCATE ,,1,13

Visualizza il cursore utente, la sua posizione rimane immutata, la forma del cursore utente e di scrittura risulta una sottile linea orizzontale al fondo della matrice dei caratteri.

LOCATE ,,1,45

Visualizza il cursore utente, la sua posizione rimane immutata, la sua forma risulta una sottile linea orizzontale al fondo della matrice dei caratteri.

Errori Possibili

Qualsiasi valore fuori dall'intervallo specificato produce l'errore "Illegal function call". I valori precedenti vengono conservati.

LOCATE (Grafica) Istruzione

Sposta il cursore nella posizione specificata. LOCATE può anche visualizzare o rendere invisibile il cursore utente e definire la sua struttura o la struttura (contemporaneamente) dei cursori utente e di scrittura. LOCATE consente anche di specificare la frequenza di intermittenza del cursore utente.

Sintassi 1

LOCATE [*row*][,[*column*][,[*rate*][,[*start*][,*stop*]]]]

Sintassi 2

LOCATE [*row*][,[*column*][,[*rate*],[*line,map*]]]

Dove

Il valore del quarto parametro determina quale tra le istruzioni riportate nella sintasssi deve essere applicata.

ELEMENTO DI SINTASSI	SIGNIFICATO
row	È il numero di linea su video; è un'espressione numerica il cui valore è un intero senza segno compreso tra 1 e 25.

ELEMENTO DI SINTASSI	SIGNIFICATO
column	È il numero di colonna su video; è un'espressione numerica il cui valore è un intero senza segno compreso tra 1 e 80, a seconda dell'ampiezza del video.
rate	È un'espressione intera compresa tra 0 e 10, che determina lo stato del cursore utente. 0 Rende invisibile il cursore utente 1 Visualizza il cursore utente. 2...10 Fa lampeggiare il cursore utente con una frequenza pari a *rate* di 1/18.75 secondi.
start	È la scanline iniziale del cursore, deve essere un'espressione intera compresa tra 0 e 15 (oppure tra 32 e 47), in Altissima Risoluzione, e tra 0 e 7 (oppure tra 32 e 39), in Media e Alta Risoluzione. Se *start* è compreso tra 0 e 15 viene programmata la forma del cursore utente e del cursore di scrittura. Se *start* è compreso tra 32 e 47 viene interpretato modulo 16 e viene programmata solo la forma del cursore utente.
stop	È la scanline finale del cursore. Deve essere un'espressione intera, con lo stesso intervallo di valori di *start* e deve essere maggiore o uguale a *start*: se uguale, il cursore si riduce ad una sola scanline.

ELEMENTO DI SINTASSI	SIGNIFICATO
line	Se il valore di *line* è compreso tra 50 e 50 + M, il numero di byte *line-50* della mappa di bit del cursore di scrittura viene posto uguale a *map*. Se il valore è compreso tra 100 e 100 + M, il numero di byte *line-100* della mappa di bit del cursore utente viene posto uguale a *map*. Il valore di M è pari a 7 in Alta Risoluzione e 15 in Media e Altissima Risoluzione.
map	Questo valore sostituisce il contenuto del byte specificato da *line*. La mappa di bit del cursore è un vettore di byte messo in XOR con i pixel presenti su video. In Media Risoluzione, ogni scanline del cursore viene rappresentata da 2 byte; infatti ogni pixel del cursore è associato a due bit contigui che specificano il colore del pixel. In Alta e Altissima Risoluzione c'è un byte per ogni scanline. Ogni modalità di grafica (Media, Alta e Altissima Risoluzione) memorizza la mappa di bit di ogni cursore (scrittura, inserimento e utente) che viene ripristinato passando da una modalità ad un'altra.

Caratteristiche

GW-BASIC include un cursore intermittente per la grafica. L'altezza massima di questo cursore è pari all'altezza della matrice dei caratteri; quindi è uguale a 8 scanline in Media e Alta Risoluzione e a 16 scanline in Altissima Risoluzione.

Le scanline del cursore vengono numerate incominciando da 0 per la scanline di inizio.

È possibile definire tre cursori diversi in uno dei modi di grafica così come nel Modo Testo.

Il cursore in stato inserimento è sempre un piccolo triangolo, con alta frequenza di intermittenza, che si trova nell'angolo in basso a sinistra della matrice dei caratteri. Il cursore di scrittura è inizialmente una sottolineatura, che lampeggia più lentamente. Il cursore utente inizialmente è disabilitato. Il vettore che descrive la forma del cursore utente e di scrittura viene inizializzato con byte FF in modo da poter facilmente acquisire la forma di una sottolineatura o di un rettangolo pieno. La forma del cursore utente e del cursore di scrittura è programmabile.

Anche la frequenza di l'ampeggiamento del cursore utente è programmabile, mentre quella dei cursori di scrittura e di inserimento è fissa.

LOCATE ,,0 rende invisibile soltanto il cursore utente. Anche l'esecuzione di qualsiasi istruzione di grafica (LINE, PSET, ecc.) rende invisibile il cursore utente.

Il cursore di scrittura viene visualizzato ad ogni esecuzione dell'istruzione INPUT e quando l'utente imposta o corregge una linea GW-BASIC. Il cursore inserimento viene visualizzato quando l'utente introduce caratteri su una linea GW-BASIC. Tutte le altre volte può comparire solo il cursore utente.

Esempio 1

```
10 LOCATE 5,1,4,5,7
```

Sposta il cursore alla linea 5, colonna 1, visualizza il cursore utente a una frequenza di intermittenza pari a 4/18.75 secondi e stabilisce l'altezza del cursore utente e di scrittura pari a 3 scanline.

Il vettore che descrive la forma del cursore viene inizializzato a FF (se l'utente non ha modificato la mappa di bit).

Esempio 2

```
100 LOCATE ,,,51,&H82
110 LOCATE ,,,103,&H01
```

Queste istruzioni specificano i byte nella mappa di bit rispettivamente per il cursore di scrittura e utente. L'istruzione 100 imposta il byte 1 del cursore di scrittura al valore esadecimale 82; l'istruzione 110 imposta il byte 3 del cursore utente al valore esadecimale 01.

Esempio 3

Questo esempio cancella la mappa di bit del cursore di scrittura in Media Risoluzione:

```
10 SCREEN 1
   .
   .
   .
50 FOR I=0 TO 15
60 LOCATE ,,,50+I,0
70 NEXT I
```

Questo esempio cancella la mappa di bit del cursore di scrittura in Alta Risoluzione:

```
10 SCREEN 2
   .
   .
   .
50 FOR I=0 TO 7
60 LOCATE ,,,50+I,0
70 NEXT I
```

Questo esempio cancella la mappa di bit del cursore di scrittura in Altissima Risoluzione:

```
10 SCREEN 3
    .
    .
    .
50 FOR I=1 TO 15
60 LOCATE ,,,50+I,0
70 NEXT I
```

LOCK Istruzione

L'istruzione LOCK limita l'accesso da parte di altri processi ad un file aperto.

L'istruzione può essere usata solo se è stato installato MS-NET con MS-DOS 3.1 o future versioni.

LOCK[#] *filenum* [,[*recordnum1*] [**TO** *recordnum2*]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filenum	È il numero del file aperto.
recordnum1	È il numero del record dal quale incomincia la protezione. Se questo parametro è omesso, viene assunto il record numero 1 come inizio della protezione.

ELEMENTO DI SINTASSI	SIGNIFICATO
recordnum2	È il numero del record dove termina la protezione. Se questo parametro è omesso, viene protetto solo il record espresso da *recordnum1*.

Caratteristiche

Se il file è stato correttamente aperto per operazioni di input/output sequenziali, la protezione riguarda ogni sua parte.

Se il file è stato aperto in modo random, l'intervallo indica i record del file che saranno protetti.

Vedere le istruzioni OPEN e UNLOCK.

Nota

Se l'utente tenta di usare questa istruzione con release MS-DOS precedenti alla 3.1, incorrerà in un "Advanced Feature Error". Se si sta usando MS-DOS 3.1 o future versioni, ma MS-NET non è stato installato, si incorrerà in un errore di "Permission Denied".

Si deve introdurre il comando SHARE di MS-DOS (o alla comparsa del prompt di MS-DOS, o in un file AUTOEXEC.BAT).

Vedere il manuale "MS-DOS Guida Utente" per una completa descrizione delle procedure da seguire per la rete.

Il modo consigliato per utilizzare file su device condivisi, è quello di eseguire un LOCK su un file o un intervallo di record all'interno di un file, prima di tentare di leggere o scrivere su quel file. È anche consigliabile eseguire un UNLOCK su quel file o intervallo di record, prima che questi venga chiuso.
Se questo non viene fatto, si potrebbero verificare dei problemi nell'accesso a quel file in futuro, in ambiente di rete.
Il tempo di LOCK dovrebbe essere mantenuto al minimo.

Esempi

LOCK #1	protegge per intero il file #1
LOCK #1,2	protegge il record 2 del file #1
LOCK #1, TO 23	protegge i record da 1 a 23 del file #1
LOCK #1, 12 TO 24	protegge i record da 12 a 24 del file #1

Errori Possibili

"Permission Denied": MS-NET non installato o tentativo di accesso ad un file i cui accessi sono limitati.

"Illegal Function Call": l'intervallo specificato non è corretto o eccede le dimensioni del file.

LOF Funzione

Restituisce il numero di byte del file specificato.

LOF(*filenum*)

Caratteristiche

Per file ad accesso diretto e sequenziale, la funzione LOF restituisce il numero di byte del file. Quando un file viene aperto per operazioni di APPEND o OUTPUT, la funzione LOF restituisce la dimensione del file divisa per 128.

Per file di comunicazione, LOF può essere usata per controllare se il buffer di input si sta riempiendo e restituisce la quantità di spazio libero presente nel buffer. Cioè:

buffer-size-**LOF** (*filenum*)

dove *buffer-size* rappresenta la dimensione del buffer di comunicazione, che assume un valore di default di 256 byte, ma può essere cambiata tramite l'opzione /C: impostando il comando GWBASIC.

Esempio 1

```
10 OPEN "B:MYFILE" AS #2
20 GET #2, LOF(1)/128
.
.
.
```

Le istruzioni precedenti leggono l'ultimo record del file MYFILE (residente sul disco inserito nel drive B), assumendo che il file sia stato creato con record di 128 byte.

Esempio 2

```
100 IF REC*RECSIZ>LOF(1) THEN PRINT "INVALID ENTRY"
```

In questo esempio le variabili REC e RECSIZE contengono rispettivamente il numero e la lunghezza del record. Il calcolo determina se il record specificato si trova oltre la fine del file.

LOG Funzione

Calcola il logaritmo naturale di un argomento positivo.

LOG(*numexp*)

Caratteristiche

Il logaritmo viene calcolato in semplice precisione, se non viene specificata l'opzione /D impostando il comando GWBASIC.

Se *numexp* è negativo o uguale a zero, si verifica l'errore "Illegal function call".

Esempio

```
PRINT LOG(45/7)
 1,860752
Ok
```

LPOS Funzione

Restituisce la posizione della testina di stampa all'interno del buffer di stampante.

La funzione LPOS viene di solito usata all'interno di un programma.

LPOS(*printer*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
printer	È un'espressione intera il cui valore (1,2, o 3) indica la stampante da controllare (LPT1:, LPT2: o LPT3:).

Osservazioni

La funzione LPOS non restituisce necessariamente la posizione fisica della testina di stampa.

Esempio

```
100 IF LPOS(1)>60 THEN LPRINT CHR$(13)
```

LPRINT e LPRINT USING Istruzioni

LPRINT Permette di stampare dati in un formato standard.

LPRINT USING permette di stampare dati in un formato definito dall'utente.

Sintassi 1

LPRINT [*list of expressions*][, I ;]

Sintassi 2

LPRINT USING *format-string* ; *list-of-expressions*[, I ;]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
list-of-expressions	È una lista di espressioni numeriche e/o stringa separate da virgola o punti e virgola.
format-string	È un'espressione stringa (di solito una costante o una variabile stringa) composta da speciali caratteri di formattazione.

Caratteristiche

Queste istruzioni si comportano come PRINT e PRINT USING. L'unica differenza è l'output diretto alla stampante. LPRINT assume una stampante con riga di 132 caratteri.

Esempio

```
10 A$ = "For July..."
20 X = .491
30 LPRINT "Results", A$,
40 LPRINT X
RUN
Results    For July...    .491
Ok
```

Il risultato compare sulla stampante.

LSET e RSET Istruzioni

LSET memorizza un valore stringa in un campo di un buffer ad accesso diretto, allineato a sinistra, o allinea a sinistra un valore stringa in una variabile stringa.

RSET memorizza un valore stringa in un campo di un buffer ad accesso diretto, allineato a destra, o allinea a destra un valore stringa in una variabile stringa.

LSET ed RSET vengono di solito usate all'interno di un programma.

Sintassi 1

LSET *stringvar* = *stringexp*

Sintassi 2

RSET *stringvar* = *stringexp*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
stringvar	Rappresenta una variabile defìnta precedentemente tramite l'istruzione FIELD.

ELEMENTO DI SINTASSI	SIGNIFICATO
stringexp	Rappresenta la stringa da allineare a sinistra o a destra in un dato campo.

Caratteristiche

Se *stringexp* richiede meno byte di quanti sono stati definiti dall'istruzione FIELD per *stringvar*, LSET allinea a sinistra la stringa nel campo, mentre RSET la allinea a destra (riempiendo con spazi). Se la stringa è più lunga del campo definito, i caratteri vengono troncati a destra. I valori numerici devono essere convertiti in valori stringa, prima di venire usati con le istruzioni LSET o RSET. Per una descrizione completa dei file ad accesso diretto si veda anche il Capitolo 4.

Esempi

```
150 LSET A$ = MKS$(AMT)
160 LSET D$ = MKI$ (COUNT$)
```

Le istruzioni LSET e RSET possono essere utilizzate con una variabile stringa non definita dall'istruzione FIELD, per l'allineamento a sinistra o a destra di una stringa in un dato campo. Per esempio le linee di programma:

```
110 A$ = SPACE$(20)
120 RSET A$ = N$
```

allineano a destra la stringa N$ in un campo di 20 caratteri. Questa può essere un'utile tecnica di formattazione per le operazioni di stampa.

MERGE Comando

Include nel programma residente in memoria le linee di un programma GW-BASIC registrato su disco, in formato ASCII.

MERGE { *filespec* I *pathname* }

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filespec (o *pathname*)	È un'espressione stringa che specifica un file sorgente GW-BASIC e, opzionalmente, il drive su cui il file risiede. Se viene omesso il drive viene assunto il drive di default di MS-DOS.

Caratteristiche

Il comando MERGE permette di includere nel programma residente in memoria, un programma specificato registrato su disco (in formato ASCII). Il comando MERGE è simile al comando LOAD, ma il programma in memoria non viene cancellato prima di

trasferire il programma da disco in memoria. Il programma su disco viene invece inserito all'interno del programma residente in memoria, in modo da formare con questo un unico programma. In altre parole le linee del programma su disco vengono semplicemente inserite tra le linee del programma residente in memoria in ordine sequenziale. Se una linea del programma su disco e una del programma residente hanno lo stesso numero di linea, la linea del programma su disco sostituisce quella del programma residente in memoria.

Esempio

```
MERGE "B:ROOT\S1\SUBRTN"
```

MID$ Funzione e Istruzione

Usata come funzione, MID$ estrae una sottostringa da una stringa specificata. Usata come istruzione, MID$ sostituisce una parte di una stringa con un'altra stringa.

Sintassi 1: funzione

MID$(*string,start*[,*length*]**)**

Sintassi 2: istruzione

MID$(*string,start*[,*length*]**)** = *substring*

Dove: Usata come Funzione

ELEMENTO DI SINTASSI	SIGNIFICATO
string	È un'espressione stringa il cui valore corrisponde alla stringa dalla quale la sottostringa deve essere estratta.
start	È un'espressione intera il cui valore specifica la posizione del carattere iniziale della stringa da estrarre e deve essere > = 1.
length	È un'espressione intera, compresa tra 0 e 255, che rappresenta la lunghezza della stringa da estrarre.

Caratteristiche

La funzione estrae dalla stringa specificata una sottostringa iniziando dal carattere situato nella posizione indicata da *start*. È possibile specificare la lunghezza (*length*) della sottostringa estratta. Se il parametro *length* viene omesso o se, alla destra del carattere situato nella posizione data, ci sono meno caratteri di quelli specificati da *length* vengono estratti tutti i caratteri più a destra a partire dal carattere specificato.

Se *length* è uguale a zero o se *start* è maggiore di LEN(*string*) la funzione estrae una stringa nulla.

Si vedano anche le funzioni LEFT$ e RIGHT$.

Dove: Usata come Istruzione

ELEMENTO DI SINTASSI	SIGNIFICATO
string	È un'espressione stringa il cui valore rappresenta la stringa di cui bisogna sostituirne una parte.
start	È un'espressione intera, compresa tra 1 e 255, il cui valore specifica la posizione del carattere da cui deve iniziare la sostituzione; il parametro *start* deve essere < = LEN(*string*).
length	È un'espressione intera, compresa tra 0 e 255, che rappresenta la lunghezza della parte di stringa da sostituire.
substring	È un'espressione stringa che sostituisce i caratteri di *string*, a partire dalla posizione *start*.

Caratteristiche

I caratteri di *string* vengono sostituiti con i caratteri di *substring* a partire dalla posizione *start*. Il parametro opzionale *length* si riferisce al numero di caratteri di *substring* che verranno usati nella sostituzione. Se viene omesso *length*, vengono usati tutti i caratteri di *substring*.

Comunque, indipendentemente dalla specificazione del parametro *length*, la sostituzione dei caratteri non supera mai la lunghezza iniziale di *string*.

Esempi

Nell'esempio seguente viene usata la funzione MID$.

```
Ok
10 A$ = "HELLO"
20 B$ = "JOSEPH JOHNNY JIMMY"
30 PRINT A$; MID$(B$,8,6)
RUN
HELLO JOHNNY
Ok
```

Nell'esempio seguente viene usata l'istruzione MID$.

```
Ok
10 A$ = "AVIGNON, FRANCE"
20 MID$(A$,9) = "ROUBAIX"
30 PRINT A$
RUN
AVIGNON, ROUBAI
Ok
```

Errori Possibili

Se *start* o *length* sono fuori dall'intervallo specificato, viene visualizzato il messaggio d'errore "Illegal function call".

MKDIR Comando

Permette la creazione di una nuova directory su di un disco specificato.

MKDIR *pathname*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
pathname	È un'espressione stringa che specifica il nome della directory che deve essere creata.

Esempi

```
        ROOT
          |
      ACCOUNTS
      /       \
  AMOS         ANDY
```

Con riferimento alla struttura ad albero riportata sopra, le istruzioni che seguono creano delle sottodirectory.

Se la directory corrente è root, per creare la sottodirectory SALES sul drive corrente, partendo da root, è necessario inserire:

MKDIR "SALES"

Per creare la sottodirectory FRED sotto la directory SALES è necessario inserire:

MKDIR "SALES \FRED"

Per creare la sottodirectory WILMA sotto la directory FRED è necessario inserire:

MKDIR "SALES\FRED\WILMA"

La struttura finale risulta:

```
                 ROOT
                /    \
          SALES       ACCOUNTS
          /           /      \
       FRED        AMOS     ANDY
       /
   WILMA
```

Errori Possibili

"Bad file name": name della directory errato.

"Path/File access error": errore di accesso a Path/File

MKI$,MKS$,MKD$ Funzioni

Convertono valori numerici in valori di tipo stringa.

Sintassi 1

MKI$(*integer-expression***)**

Sintassi 2

MKS$(*single-precision-expression***)**

Sintassi 3

MKD$(*double-precision-expression***)**

Caratteristiche

Qualsiasi valore numerico trasferito nel buffer di un file ad accesso diretto tramite un'istruzione LSET o RSET deve essere convertito in un valore di tipo stringa. MKI$ converte un'intero in una stringa di 2-byte, MKS$ converte un numero in semplice precisione in una stringa di 4-byte e MKD$ converte un numero in doppia precisione in una stringa di 8 byte.

Si vedano anche le funzioni CVI, CVS, CVD.

Esempio

```
90 AMT=(K+T)
100 FIELD #1,8 AS D$,20 AS N$
110 LSET D$=MKS$(AMT)
120 LSET N$=A$
130 PUT #1
.
.
```

NAME Comando

Cambia il nome di un file su disco.

Sintassi 1

NAME {*filespec* | *pathname*} **AS** *filename*

Sintassi 2

NAME {*filespec* | *pathname*} **AS** *new-pathname*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filespec (o *pathname*)	È un'espressione stringa che specifica il file il cui nome deve essere modificato. Il file deve esistere sul drive specificato. Se il drive non viene specificato, viene assunto il drive di default di MS-DOS. L'estensione del file non assume il default .BAS.
filename	È il nuovo nome del file che sostituisce quello precedente; non deve essere usato per un altro file.
new-pathname	È un'espressione stringa che specifica la nuova directory del file, quando viene usato il comando NAME per spostare un file da una directory a un'altra.

Caratteristiche

Dopo il comando NAME il file continua a esistere sullo stesso disco e nella stessa area, ma con il nuovo nome. Anche l'area allocata per il file non cambia. Non è possibile cambiare il nome

di un file con una nuova designazione nel drive; nel caso si tenti di fare ciò, si verifica l'errore "Rename across disks". Prima di eseguire il comando NAME, il file deve essere chiuso.

Il comando NAME può anche essere usato per spostare un file da una directory ad un'altra. Per esempio:

NAME "\X\CLIENTS" AS "\XYZ\P\CLIENTS"

Esempio

Ok
NAME "B:GRAPH.BAS" AS "GRAPH1.BAS"
Ok

In questo esempio il file chiamato precedentemente GRAPH.BAS sul disco del drive B: viene ora chiamato GRAPH1.BAS.

Errori Possibili

"File not found": file inesistente.

"File already exists": il nome specificato è identico a quello di un file esistente su disco.

"Bad file name": nome di file errato.

"Too many files": directory già satura.

"Rename across disks": nome, dichiarato su un disco, diverso da quello specificato per il nome precedente.

NEW Comando

Cancella il programma corrente e azzera tutte le variabili, permettendo l'introduzione di un nuovo programma.

NEW

Caratteristiche

Il comando NEW viene digitato in stato comandi per azzerare la memoria prima di introdurre un nuovo programma. GW-BASIC ritorna sempre in stato comandi dopo l'esecuzione di tale comando. NEW effettua la chiusura di tutti i file dati e disabilita il flag di trace con la stessa modalità del comando TROFF.

OCT$ Funzione

Restituisce una stringa contenente il valore ottale di un numero decimale.

OCT$(*numexp*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
numexp	È un'espressione numerica compresa fra -32768 e 65535, il cui valore viene arrotondato all'intero più vicino prima di essere calcolato dalla funzione OCT$.

Osservazioni

Se *numexp* è un numero negativo viene usato il complemento a due.

Esempio

```
PRINT OCT$(24)
 30
Ok
```

Per dettagli sulla conversione esadecimale si veda la funzione HEX$.

ON COM(*n*) GOSUB Istruzione

Indica il numero di linea del sottoprogramma a cui deve essere trasferito il controllo non appena viene rilevata la presenza di caratteri nel buffer di comunicazione.

L'istruzione ON COM(*n*) GOSUB deve essere usata solo all'interno di un programma.

ON COM(*n*) GOSUB *linenum*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È il numero del canale di comunicazione (1 o 2).
linenum	È il numero della prima linea della routine di rilevamento errori. Se tale numero di linea è uguale a zero il rilevamento dell'attività di comunicazione viene disabilitato.

Per Abilitare o Disabilitare il Rilevamento COM

SE...	ALLORA...
viene eseguita l'istruzione COM(*n*) ON	viene abilitato il rilevamento COM(*n*).
viene eseguita l'istruzione COM(*n*) OFF	viene disabilitato il rilevamento COM(*n*).
viene eseguita l'istruzione COM(*n*) STOP	il rilevamento COM(*n*) viene sospeso, cioé l'istruzione GOSUB non viene eseguita fino a quando non viene eseguita un'istruzione COM(*n*) ON.

SE...	ALLORA...
viene eseguita l'istruzione ON COM(*n*) GOSUB	viene sospeso il rilevamento COM(*n*).
si verifica un errore	tutte le funzioni di rilevamento di eventi vengono disabilitate (compreso il rilevamento di errori).

Per Attivare la Routine di Rilevamento COM(*n*)

SE...	ALLORA...
viene abilitato il rilevamento COM(*n*), e sono presenti dei caratteri nel buffer di COM	viene eseguita l'istruzione ON COM(*n*) GOSUB *linenum* e viene attivata la corrispondente routine. Per evitare l'esecuzione automatica di rilevamenti succesivi, viene eseguita automaticamente l'istruzione COM(*n*) STOP, dopo il rilevamento del primo errore. Dopo l'esecuzione dell'istruzione RETURN della routine di rilevamento, viene automaticamente eseguita un'istruzione COM(*n*) ON (se non è già stata eseguita una istruzione COM(*n*) OFF all'interno della routine di rilevamento). Può anche essere usata l'istruzione RETURN *linenum*, facendo però attenzione perché eventuali istruzioni GOSUB, WHILE o FOR rimangono in tal caso attive generando condizioni di errore (per esempio "FOR without NEXT")

La funzione tipica della routine di rilevamento COM è quella di leggere un intero messaggio dalla porta di COM prima di restituire il controllo al programma principale. La routine di rilevamento COM non deve essere usata per leggere messaggi di un solo carattere poiché, ad elevate velocità di trasmissione, il tempo necessario per il rilevamento e la lettura di ogni singolo carattere può provocare il superamento della capacità del buffer di COM.

Esempio

In questo esempio viene attivata una routine di rilevamento per il secondo canale di comunicazione alla linea 1000.

```
100 ON COM(2) GOSUB 1000
110 COM(2) ON
 .
 .
 .
1000 REM COM è attivata
 .
 .
 .
1050 RETURN 200
```

ON ERROR GOTO Istruzione

Abilita il rilevamento di errori specificando il numero della prima linea del sottoprogramma da eseguire in caso di errore.

L'istruzione ON ERROR GOTO è usata solo all'interno di un programma.

ON ERROR GOTO *linenum*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
linenum	È il numero della prima linea della routine di rilevamento errori.

Per Abilitare o Disabilitare il Rilevamento di Errori

SE...	ALLORA...
viene eseguita l'istruzione ON ERROR GOTO *n*	se n è diverso da zero, viene abilitato il rilevamento di errori.
viene eseguita l'istruzione ON ERROR GOTO 0	viene disabilitato il rilevamento di errori. Se si verifica un'errore dopo la disabilitazione della routine, viene visualizzato il messaggio d'errore associato e l'esecuzione del programma viene interrotta.

Per attivare la Routine di Rilevamento ERROR

SE...	ALLORA...
viene abilitato il rilevamento di errori e viene rilevato un errore GW-BASIC	viene eseguita l'istruzione ON ERROR GOTO *linenum* e viene attivata la corrispondente routine. Le funzioni ERL e ERR

SE...	ALLORA...
(o un errore utente)	vengono di solito usate in istruzioni del tipo IF...GOTO...ELSE o IF...THEN...ELSE per trasferire il controllo in punti diversi all'interno della routine di rilevamento di errori. È consigliabile eseguire un'istruzione ON ERROR GOTO 0 all'interno della routine di rilevamento di errori nel caso in cui venga rilevato un errore per il quale non è prevista nessuna azione di recovery (in tal caso viene visualizzato il messaggio di errore standard e l'esecuzione del programma viene interrotta). Dopo l'interruzione della routine di gestione dell'errore, l'esecuzione può essere riattivata tramite l'istruzione RESUME.
viene rilevato un errore GW-BASIC (o un errore utente) durante l'esecuzione di una routine di rilevamento di errori	viene visualizzato il messaggio di errore associato e l'esecuzione termina. Se ha luogo un rilevamento di errori, vengono disabilitate tutte le routine di rilevamento.

Esempio

```
10 ON ERROR GOTO 100
20 INPUT R
30 IF R < =0 THEN ERROR 300
 .
 .
 .
```

```
100 IF ERR = 300 THEN PRINT "RADIUS NEGATIVE OR ZERO"
110 IF ERL = 30 THEN RESUME 20
120 ON ERROR GOTO 0
.
.
.
```

ON...GOSUB e ON...GOTO Istruzioni

ON...GOSUB richiama un sottoprogramma scelto fra un certo numero di sottoprogrammi specificati, a seconda del valore di una data espressione.

ON...GOTO trasferisce il controllo a una linea scelta fra un insieme di linee specificate, a seconda del valore di una data espressione.

Le istruzioni ON...GOSUB e ON...GOTO vengono usate all'interno di un programma.

Sintassi 1

ON *numexp* **GOSUB** *linenum*[,*linenum*]...

Sintassi 2

ON *numexp* **GOTO** *linenum*[,*linenum*]...

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
numexp	È un'espressione numerica, compresa tra 0 e 255, che specifica il numero della linea a cui viene trasferito il controllo. Se *numexp* non è un intero, esso viene arrotondato per eccesso all'intero più vicino.
linenum	È il numero della linea a cui viene trasferito il controllo.

Caratteristiche

Nell'istruzione ON...GOSUB, ogni numero di linea della lista deve essere il primo numero di linea di un sottoprogramma. Se il valore di *numexp* è uguale a 1 viene richiamato il sottoprogramma che si trova al primo numero di linea dell'elenco; se il valore è uguale a 2 viene richiamato il sottoprogramma al secondo numero di linea e così via.

Se il valore di *numexp* è uguale a zero o maggiore del numero di elementi della lista (ma minore o uguale a 255) viene eseguita la successiva istruzione. Se il valore di *numexp* è negativo o maggiore di 255 si ha il messaggio di errore "Illegal function call".

Esempio

```
100 ON L-1 GOTO 150,300,320,390
```

ON KEY(*n*) GOSUB Istruzione

Specifica la prima linea di un sottoprogramma che deve essere eseguito quando viene premuto un tasto specificato.

L'istruzione ON KEY(*n*) è usata solo all'interno un programma.

ON KEY(*n*) GOSUB *linenum*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È un intero compreso nell'intervallo da 1 a 20 che indica quale tasto deve essere rilevato cioè: 1-10 Tasti funzione da **F1** a **F10** 11 Cursor Up 12 Cursor left 13 Cursor Right 14 Cursor Down 15-20 Tasti definiti nella forma: KEY(*n*), CHR$ (*shift*) + CHR$ (*scan-code*).
linenum	È il numero di linea di inizio della routine di rilevamento errori associata al tasto specificato. Se *linenum* è uguale a zero la funzione di rilevamento associata al tasto è disabilitata.

Per Abilitare o Disabilitare il Rilevamento di un Tasto

SE...	ALLORA...
viene eseguita l'istruzione KEY(*n*) ON	viene abilitato il rilevamento del tasto *n*.
viene eseguita l'istruzione KEY(*n*) OFF	viene disabilitato il rilevamento del tasto *n*.
viene eseguita l'istruzione KEY(*n*) STOP	viene sospeso il rilevamento del tasto *n*, cioé l'istruzione GOSUB viene eseguita solo dopo un'istruzione KEY(*n*) ON.
viene eseguita l'istruzione ON KEY(*n*) GOSUB	viene sospeso il rilevamento del tasto *n*.
viene rilevato un errore	tutte le routine di rilevamento vengono disabilitate.

Per Attivare la Routine di Rilevamento KEY(*n*)

SE...	ALLORA...
viene abilitato il rilevamento del tasto *n* ed è stato premuto il tasto n	viene eseguita l'istruzione ON KEY(*n*) GOSUB e viene attivata la routine corrispondente.

▶

SE...	ALLORA...
	Per evitare ulteriori rilevamenti relativi ad uno stesso tasto, viene eseguita automaticamente l'istruzione KEY(*n*) STOP. Quando viene incontrata l'istruzione RETURN viene eseguita automaticamente l'istruzione KEY(*n*) ON (se non è stata eseguita un'istruzione KEY(*n*) OFF all'interno della routine di rilevamento). Può anche essere usata la forma RETURN *linenum*, facendo però molta attenzione perché eventuali GOSUB, WHILE o FOR attive rimangono attive e possono quindi dare luogo a errori.

Osservazioni

Quando viene effettuato il rilevamento di un tasto, l'evento che lo ha generato non viene conservato, perciò non è possibile usare successive istruzioni INPUT o INKEY$ per scoprire il tasto che ha causato il rilevamento. Quindi, per assegnare funzioni diverse a certi tasti, è necessario associare un sottoprogramma diverso a ogni tasto, piuttosto che usare un solo sottoprogramma contenente le varie funzioni associate ai tasti.

Esempio 1

```
10 KEY 4,"SCREEN 0,0" 'assegna al tasto 4 la funzione soft key
20 KEY(4) ON          'abilita il rilevamento di eventi
.
.
.
```

```
70 ON KEY(4) GOSUB 200
.
.
.
viene premuto il tasto 4
.
.
.
200 'Sottoprogramma per il video
.
.
.
250 RETURN
```

In questo esempio è stata modificata la normale funzione associata al tasto funzione 4 sostituendola con "SCREEN 0,0", visualizzato ogni volta che viene premuto il tasto. Il valore originario può essere riassegnato e riprendere quindi la sua funzione standard quando il sistema viene reinizializzato.

Esempio 2

```
100 KEY 15, CHR$(&H04) + CHR$(83)
105 REM ** il tasto 15 è ora CTRL DEL **
110 KEY(15) ON
.
.
.
1000 PRINT "If you want to stop processing for a break"
1010 PRINT "press the CTRL key and DEL at the"
1020 PRINT "same time."
1030 ON KEY(15) GOSUB 3000.
.
 L'utente preme CTRL DEL
.
3000 REM ** Sospende l'esecuzione del programma.
3010 CLOSE  1
3020 RESET
3030 CLS
```

```
3035 PRINT "Enter CONT to continue."
3040 STOP
3050 OPEN "A", #1, "ACCOUNTS.DAT"
3060 RETURN
```

In questo esempio è stato abilitato il tasto **CTRL DEL**, in modo tale da impostare un sottoprogramma che chiude i file ed interrompe l'esecuzione del programma finché l'operatore non è pronto a continuare.

ON PLAY(*n*) GOSUB Istruzione

Specifica la prima linea di un sottoprogramma che deve essere eseguito quando il buffer di musica contiene meno di *n* note. Ciò permette la continuità della musica di fondo durante l'esecuzione del programma.

L'istruzione ON PLAY(*n*) GOSUB viene usata solo all'interno di un programma.

ON PLAY(*n*) GOSUB *linenum*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È un'espressione intera compresa tra 1 e 32. I valori non compresi in questo intervallo generano l'errore "Illegal function call".

ELEMENTO DI SINTASSI	SIGNIFICATO
linenum	È il primo numero di linea della routine di rilevamento associata. Il numero di linea 0 disabilita il rilevamento del suono.

Per Abilitare o Disabilitare il Rilevamento PLAY(*n*)

SE...	ALLORA...
viene eseguita l'istruzione PLAY ON	viene abilitato il rilevamento PLAY(*n*).
viene eseguita l'istruzione PLAY OFF	viene disabilitato il rilevamento PLAY(*n*).
viene eseguita l'istruzione PLAY STOP	il rilevamento PLAY(*n*) viene sospeso, cioé la clausola GOSUB non viene eseguita ma viene eseguita subito dopo la clausola PLAY ON.
viene eseguita l'istruzione ON PLAY(*n*) GOSUB	viene sospeso il rilevamento PLAY(*n*).
si verifica un errore	tutte le routine di rilevamento vengono disabilitate.

Per Attivare una Routine di Rilevamento PLAY(*n*)

SE...	ALLORA...
viene abilitato il rilevamento PLAY(*n*), ed il buffer per la musica di background è passato da *n* a *n-1* note	viene eseguita la linea ON PLAY(*n*) GOSUB, e attivata la routine corrispondente. Per evitare il ripetersi di rilevamenti, quando si verifica il rilevamento viene automaticamente eseguito un comando PLAY STOP. Un comando RETURN dal sottoprogramma di rilevamento esegue automaticamente PLAY(*n*) ON (se all'interno della routine di rilevamento non è stata eseguita l'istruzione PLAY(*n*) OFF). Può anche essere usata la forma RETURN *linenum*. Occorre usare questa forma con attenzione, perché qualsiasi altra istruzione GOSUB, WHILE o FOR attiva rimane attiva e possono verificarsi degli errori (come "FOR without NEXT").
il programma è in corso di esecuzione, il rilevamento PLAY(*n*) è abilitato e il buffer per la musica di background è vuoto	nessuna routine di rilevamento PLAY(*n*) viene eseguita.

Osservazioni

1. Il rilevamento degli eventi per l'istruzione PLAY è efficace solo quando è in funzione Music Background (PLAY "MB..."). Le routine di rilevamento degli eventi non hanno alcun effetto se eseguite in Music Foreground (PLAY "MF...").

2. Una routine di rilevamento di eventi per l'istruzione PLAY non ha alcun effetto se il buffer di Music Background è già vuoto quando viene eseguita PLAY ON.

3. Bisogna fare attenzione nella selezione dei valori di *n*. Se *n* è un numero grande, il rilevamenento di eventi si verifica con una frequenza sufficiente da ridurre la velocità di esecuzione del programma.

Esempio

```
10 PLAY ON
20 ON PLAY(8) GOSUB 1000
 .
 .
 .
1000 'SUBROUTINE DI RILEVAMENTO PLAY(8)
1050 RETURN
```

ON TIMER(*n*) GOSUB Istruzione

Provoca un rilevamento di eventi ogni *n* secondi.

L'istruzione ON TIMER(*n*) GOSUB viene usata solo all'interno di un programma.

ON TIMER(*n*) GOSUB *linenum*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È un'espressione intera compresa fra 1 e 86400 (da 1 secondo a 24 ore). I valori fuori da questi limiti generano l'errore "Illegal function call".
linenum	È la prima linea del sottoprogramma di rilevamento eventi TIMER che deve essere eseguito.

Caratteristiche

L'istruzione ON TIMER(*n*) GOSUB viene eseguita solo se una istruzione TIMER ON è stata eseguita per abilitare un rilevamento di eventi. Se il rilevamento di eventi è abilitato e se *linenum*, nell'istruzione ON TIMER(*n*) GOSUB, è diversa da zero, GW-BASIC controlla tra le istruzioni per vedere se il tempo è stato raggiunto. In tal caso viene eseguita GOSUB alla linea specificata.

Se è stata eseguita l'istruzione TIMER OFF, GOSUB non viene eseguita e non è memorizzata.

Se è stata eseguita l'istruzione TIMER STOP, GOSUB non viene eseguita, ma verrà eseguita subito dopo l'istruzione TIMER ON.

Se viene eseguita l'istruzione ON TIMER(*n*) GOSUB, viene eseguita un'istruzione TIMER STOP automatica, in modo da evitare il verificarsi di ulteriori rilevamenti. RETURN dal sottoprogramma di rilevamento esegue automaticamente un'istruzione TIMER ON, se non è stata eseguita un'istruzione TIMER OFF all'interno del sottoprogramma.

Per ritornare a una specifica linea del sottoprogramma di rilevamento può essere usata la forma RETURN *linenum*. Occorre usare con molta attenzione questa forma, poiché qualsiasi altra istruzione GOSUB, WHILE o FOR, attiva al momento del rilevamento, rimane attiva e può generare errori come "FOR without NEXT".

Esempio

Per visualizzare l'ora del giorno sulla linea 1 ad ogni minuto

```
10 ON TIMER(60) GOSUB 1000
20 TIMER ON
 .
 .
 .
1000 OLDROW = CSRLIN 'salva la riga corrente
1010 OLDCOL = POS(0)     'salva la colonna corrente
1020 LOCATE 1,1: PRINT TIME$
1030 LOCATE OLDROW, OLDCOL 'ristabilisce riga e colonna
1040 RETURN
```

OPEN Istruzione

Permette operazioni di I/O su un file o su un device. Viene usata all'interno di un programma.

Sintassi 1

OPEN {*device* | *filespec*| *pathname*} [**FOR** *mode1*] [*access*]
AS [#]*filenum* [**LEN** = *record-length*]

Sintassi 2

OPEN *mode2* , [#]*filenum*,{*filespec* | *pathname*} [,*record-length*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
device	È un'espressione stringa che specifica il device da aprire.
filespec	È un'espressione stringa che specifica il file da aprire. Può anche specificare il device.
pathname	È un'espressione stringa che specifica il file da aprire. Può anche specificare il device.
mode1	È una stringa di caratteri, non racchiusa tra doppi apici, che determina la posizione iniziale del puntatore al file e l'azione da intraprendere se il file non esiste. I modi e le azioni possibili sono: INPUT Specifica il modo di input sequenziale. Posiziona il puntatore all'inizio di un file esistente. Se il file non esiste, viene visualizzato il messaggio d'errore "File not found".

▶

ELEMENTO DI SINTASSI	SIGNIFICATO
	OUTPUT Specifica il modo di output sequenziale. Posiziona il puntatore all'inizio del file. Se il file non esiste, ne viene creato uno nuovo. APPEND Specifica l'output sequenziale dopo l'ultimo record del file. Posiziona il puntatore alla fine del file. Se il file non esiste, ne viene creato uno. Questo modo è valido solo per file su disco. Se viene omessa la clausola FOR *mode1*, la posizione di partenza è l'inizio del file. Se il file non esiste, ne viene creato uno. Questo viene detto I/O ad Accesso Diretto, cioé i record possono essere letti o scritti in qualsiasi posizione all'interno del file.
access	Questo parametro è usato con sistemi che abbiano installato MS-NET (MS-DOS 3.1 o release future); specifica il corretto accesso al file. I possibili accessi possono essere: SHARED Qualsiasi processo su qualsiasi macchina può leggere o scrivere il file.

ELEMENTO DI SINTASSI	SIGNIFICATO
	LOCK READ — Nessun altro processo è abilitato all'accesso in lettura su quel file; questo avviene solo se nessun altro processo ha un accesso LOCK READ sul file. LOCK WRITE — Nessun altro processo è abilitato all'accesso in scrittura su quel file; questo avviene solo se nessun altro processo ha un accesso LOCK WRITE sul file. LOCK READ WRITE — Nessun altro processo è abilitato all'accesso in lettura e scrittura su quel file. Questo avviene solo se nessun altro processo ha un accesso LOCK READ o WRITE sul file. Se l'accesso non è specificato, il file può essere aperto in lettura o scrittura dal processo, ma gli altri processi non potranno accedervi fino a che il file resta aperto. Vedere le istruzioni LOCK e UNLOCK. Notare che se l'utente tenta di usare questo parametro con una release precedente alla 3.1 di MS-DOS incorrerà in un "Advanced Feature Error".

ELEMENTO DI SINTASSI	SIGNIFICATO
	Se si sta usando MS-DOS 3.1 o release future, ma non è stato installato MS-NET, si incorrerà in un errore del tipo: "Permission Denied". Si deve introdurre il comando SHARE di MS-DOS (o alla comparsa del prompt di MS-DOS o in un file AUTOEXEC.BAT). Vedere il manuale "MS-DOS Guida Utente" per una completa descrizione delle procedure da seguire per la rete.
filenum	È un'espressione intera che restituisce un numero nell'intervallo da 1 a 255. Il numero viene usato per associare un buffer di I/O con un file su disco o un device. Questa associazione continua finché non viene eseguita una istruzione CLOSE o CLOSE *filenum*. Il numero di file viene usato per specificare il file in ogni istruzione di I/O.
record-length	È un'espressione intera compresa nell'intervallo da 2 a 32767. Questo valore specifica la lunghezza di record da usare per i file ad accesso diretto.

ELEMENTO DI SINTASSI	SIGNIFICATO
	Se viene omessa, la lunghezza del record assume il valore di default di 128 byte. Il parametro *record-length* non può superare il valore specificato dallo switch /S: nel comando GWBASIC. GW-BASIC ignora questa opzione se viene usata per aprire un file sequenziale.
mode2	È un'espressione stringa il cui carattere iniziale è uno dei seguenti: O specifica output sequenziale. I specifica intput sequenziale. R specifica input/output ad accesso diretto. A specifica output sequenziale, posiziona il puntatore alla fine del file e specifica il numero di record come ultimo record del file. Quindi una istruzione PRINT# o WRITE# estende il file.

Caratteristiche

Prima di poter eseguire eventuali operazioni di I/O su un file su disco, questo deve essere stato aperto. L'istruzione OPEN alloca un buffer per operazioni di I/O su file o su un device e determina il modo di accesso al buffer. Il parametro *filenum* specifica il numero che viene associato al file per tutto il tempo in cui il file resta aperto e viene usato per specificare il file o il device in ogni successiva istruzione di I/O.

Il massimo numero di file che possono venire aperti simultaneamente, durante l'esecuzione di un programma GW-BASIC, viene specificato dallo switch /F: nel comando GWBASIC. Questo numero è compreso tra 1 e 15 e assume il valore di default 3.

Sono leciti i seguenti modi per aprire un device o un file:

KYBD:	solo INPUT
SCRN:	solo OUTPUT
LPT1:	OUTPUT o accesso diretto (*)
LPT2:	OUTPUT o accesso diretto (*)
LPT3:	OUTPUT o accesso diretto (*)
COM1:,COM2:	INPUT o OUTPUT
File su disco	INPUT, OUTPUT, APPEND o accesso diretto

(*) GW-BASIC non invia un carattere di avanzamento riga dopo ogni **CR**, per una stampante ad accesso diretto con un'ampiezza di 255 caratteri.

Il sistema di I/O sui file GW-BASIC permette di sfruttare i device installati dall'utente.

I Character Device vengono aperti e usati esattamente come i file su disco. Comunque, i caratteri non vengono inseriti nel buffer dal GW-BASIC come per i file su disco. La lunghezza viene fissata pari a uno.

GW-BASIC invia solamente **CR** (ritorno carrello X'0D') come fine linea. Se il device richiede **LF** (avanzamento riga X'0A'), è il driver che deve fornirlo.

Quando si scrive un driver di device, si deve ricordare che gli utenti GW-BASIC vogliono leggere e scrivere informazioni di controllo. La scrittura e la lettura di dati di controllo del device vengono gestite dall'istruzione IOCTL e dalla funzione IOCTL$(*f*).

Osservazioni

1. L'impostazione di eventuali valori fuori dall'intervallo dato provoca l'errore "Illegal function call". Il file non viene aperto.

2. Se il file viene aperto per operazioni di INPUT, eventuali tentativi di scrittura sul file provocano l'errore "Bad file mode". Se il file aperto per l'input non esiste si verifica l'errore "File not found".

3. Quando un file su disco viene aperto per operazioni di APPEND, il puntatore viene inizialmente posizionato alla fine del file e il nuovo record viene posto dopo l'ultimo record del file. Le istruzioni PRINT# o WRITE# permettono quindi di estendere il file.

4. Se il file viene aperto per operazioni di OUTPUT o APPEND, eventuali tentativi di leggere il file provocano l'errore "Bad file mode".

5. Se viene aperto un file che non esiste per operazioni di output, append o per accesso diretto ne viene creato uno nuovo.

6. È possibile in qualsiasi momento aprire un certo file su disco per input sequenziale o accesso diretto, assegnando al file un numero diverso per ogni tipo di accesso.

   Un file non può essere aperto per operazioni di OUTPUT o di APPEND con più di un numero di file contemporaneamente.

   Inoltre, poiché è possibile con cammini diversi fare riferimento allo stesso file in una sottodirectory, GW-BASIC non è in grado di riconoscere se si tratta dello stesso file, considerando soltanto il cammino. Per questo motivo GW-BASIC non permette di aprire un file per operazioni di OUTPUT e APPEND se esso si trova sullo stesso disco, anche se sono specificati cammini diversi.

Esempio 1

```
10 OPEN "I",2,"INVEN"
20 OPEN "MAILING.DAT"FOR APPEND AS 1
30 OPEN "CON:"FOR INPUT LOCK READ AS#1
40 OPEN "MYFILE" FOR OUTPUT LOCK WRITE AS#2
```

Se viene scritto e installato un device chiamato F01, l'istruzione OPEN può apparire in questa forma:

```
100 OPEN "\DEV\F01" FOR OUTPUT AS #1
```

Per aprire la stampante per operazioni di output, è possibile usare la linea:

```
200 OPEN "LPT1:" FOR OUTPUT AS #1
```

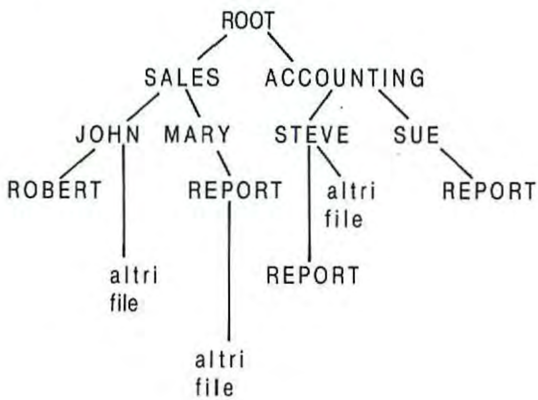
che usa il driver di device GW-BASIC, o l'elemento di pathname:

```
300 OPEN " \DEV \LPT1" FOR OUTPUT AS #1
```

che usa il driver di device MS-DOS.

Esempio 2

Usando la seguente struttura ad albero:



Se MARY è la directory corrente:

```
10 OPEN "REPORT"...
20 OPEN "\SALES\MARY\REPORT"...
30 OPEN "\MARY\REPORT"...
40 OPEN "..\.. MARY\REPORT"...
```

fanno tutte riferimento allo stesso file.

Errori Possibili

"Bad file name": nome di file non valido.

"Bad file number": numero di file non valido.

"Bad file mode": metodo di accesso non valido.

"Too many files": troppi file sono aperti (vedere lo switch /F: nella linea di comando GWBASIC).

"File not found": questo errore si verifica se viene aperto, per operazioni di input, un file che non esiste.

"Device not available": in questo caso si è tentato di aprire una directory o un device non esitente.

"File already open": file già aperto.

"Device I/O error": errore di ricezione causato, di solito, da un driver di device (installato dall'utente) scritto non correttamente.

"Illegal function call": errore causato, di solito, da un record troppo lungo (vedere lo switch /S: nella linea di comando GWBASIC).

"Permission Denied": tentativo fallito di apertura del file causato dall'accesso limitato.

"Path/File Access Error": tentativo fallito di apertura del file causato dal modo di accesso specificato che è incompatibile con l'accesso condiviso al device specificato nella rete installata.

OPEN COM Istruzione

Apre ed inizializza una porta RS-232-C per operazioni di input/output.

OPEN "COM*n*:[*speed*][,[*parity*][,[*data*][,[*stop*] [,**RS**][,**CS**[*t*]] [,**DS**[*t*]][,**CD**[*t*]][,**BIN**][,**ASC**][,**LF**]]]]"[**FOR** *mode*]**AS**[#]*filenum* [**LEN** = *record-length*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È 1 o 2 e specifica il numero della porta RS-232-C.
speed	È una costante intera che specifica la velocità di trasmissione/ricezione (in baud), in bits al secondo, relativa alla porta specificata. Può assumere i valori: 75, 110, 150, 300, 600, 1200, 1800, 2400, 4800 e 9600. Il valore di default è 300 bps.
parity	Specifica la parità. Può assumere i valori:

ELEMENTO DI SINTASSI	SIGNIFICATO
	E (pari)-valore di default M (mark) N (nessuno) O (dispari) S (spazio).
data	Specifica il numero di bit di dati per byte. Può assumere i valori: 5, 6, 7 (valore di default) o 8.
stop	Indica il numero di bit di stop. Può assumere i valori: 1, 1.5 o 2. Il valore di default è di due bit di stop per velocità pari a 75 e 110 bps, un bit di stop per tutti gli altri casi.
RS	Sopprime il segnale RTS (Request To Send).
CS[*t*]	Controlla il segnale CTS (Dear To Send).
DS[*t*]	Controlla il segnale DSR (Data Set Ready).
CD[*t*]	Controlla il segnale CD (Carrier Detect).
BIN	Apre il file in modo binario. Questo

ELEMENTO DI SINTASSI	SIGNIFICATO
	è il modo di default se non è stato specificato il parametro ASC.
ASC	Apre il file in modo ASCII.
LF	Specifica che deve essere inviato un carattere di avanzamento riga dopo **CR**.
mode	È una delle seguenti espressioni stringa: OUTPUT specifica il modo di output sequenziale. INPUT specifica il modo di input sequenziale. Se viene omessa l'espressione *mode*, viene assunto che si tratti di input/output ad accesso diretto. Il modo ad accesso diretto non può comunque essere scelto esplicitamente come *mode*.
filenum	È il numero del file da aprire.
record-length	È la lunghezza dei record scritti o letti da un buffer di comunicazione. Questo valore non può essere maggiore del valore specificato dallo switch /C nel comando GWBASIC. Il valore di default per il buffer di ricezione è 256 byte. La lunghezza del buffer di trasmissione è 128 byte.

Caratteristiche

Prima di usare una porta RS-232-C, deve essere eseguita l'istruzione OPEN COM.

Eventuali errori di sintassi nell'istruzione OPEN COM provocano l'emissione del messaggio "Bad file name".

Le opzioni *speed*, *parity*, *data* e *stop* devono essere elencate nell'ordine indicato nella sintassi del comando. Le restanti opzioni possono venire elencate in qualsiasi ordine, ma in ogni caso dopo quelle precedenti.

Le opzioni CS, DS e CD permettono di specificare un tempo (*t*) di attesa per il segnale prima di restituire l'errore "Device timeout". Questo tempo viene espresso in millesimi di secondo, da 0 a 65535. I valori di default sono: CD = 0, CS = 1000, DS = 1000.

LF permette di stampare file di comunicazione su una stampante di linea seriale. Se viene specificato LF, viene automaticamente inviato un carattere di avanzamento riga (0AHex.) dopo ogni carattere di ritorno a capo (0DHex.). Ciò include l'invio del ritorno a capo come risultato della definizione dell'ampiezza. Si noti che le istruzioni INPUT# e LINE INPUT#, se usate per leggere da un file COM aperto con l'opzione LF, terminano dopo **CR**, ignorando il comando di avanzamento riga.

L'opzione LF viene sostituta dall'opzione BIN.

Nel modo BIN, i comandi di tabulazione non sono tradotti in spazi, non viene forzato **CR** alla fine di ogni linea e **CTRL Z** non viene considerato come "end-of-file". Se il file viene chiuso, **CTRL Z** non viene inviato sulla linea RS-232. L'opzione BIN sostituisce l'opzione LF.

In modo ASC, i comandi di tabulazione sono espansi, viene forzato **CR** alla fine di ogni linea, **CTRL Z** viene considerato come "end-of-file" e viene abilitato il protocollo XON/XOFF. Se il canale è chiuso, **CTRL Z** viene inviato sulla linea RS-232.

Esempio

```
10 OPEN "COM1:9600,N,8,1,BIN" AS #2
```

Viene aperta la porta di comunicazione 1 in modo ad accesso diretto a una velocità di 9600 baud, nessun bit di parità, 8 bit di dati e 1 bit di stop. Per le operazioni di input/output viene usato il modo binario. Altre linee del programma possono ora accedere alla porta 1 come file numero 2.

OPTION BASE Istruzione

Definisce il valore minimo degli indici delle matrici.

OPTION BASE *n*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È un'espressione intera uguale a 1 o a 0.

Caratteristiche

Il valore minimo di default è uguale a 0. Se viene eseguita l'istruzione:

OPTION BASE 1

il valore minimo degli indici delle matrici è 1.

Un programma concatenato può avere un'istruzione OPTION BASE se non gli vengono passate delle matrici. Il programma concatenato assume il valore OPTION BASE del programma concatenante.

Errori Possibili

L'istruzione OPTION BASE deve essere eseguita prima di definire o usare matrici. Si verifica l'errore "Duplicate definition" se il valore minimo viene cambiato dopo aver definito una matrice tramite DIM o aver fatto riferimento ad un elemento di matrice nel programma.

OUT Istruzione

Trasmette un byte a una porta di output.

OUT *port,byte*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
port	È un'espressione intera compresa tra 0 e 65535 e rappresenta il numero di una porta di output.
byte	È un'espressione intera compresa tra 0 e 255 e rappresenta i dati da trasmettere.

Caratteristiche

L'istruzione OUT è complementare alla funzione INP.

Se *port* o *byte* si trovano fuori dall'intervallo specificato, si verifica l'errore "Illegal function call".

Esempio

```
100 OUT 1234,255
```

PAINT Istruzione

Colora l'area di una figura.

PAINT [STEP] (*x*, *y*) [,[*paint*][,[*border*][,*background*]]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
x,y	Sono le coordinate, assolute o relative, del punto da cui iniziare la colorazione. La colorazione deve iniziare sempre in un punto non appartenente al contorno; se inizia da un punto interno alla figura viene colorata la figura delimitata; se inizia da un punto esterno viene colorata l'area esterna alla figura fino ai limiti della finestra fisica corrente.
paint	È un'espressione numerica o stringa. Se è un'espressione numerica deve essere compresa tra 0 e 3 e rappresenta il numero di colore che deve essere usato. In Media Risoluzione, *paint* sceglie il colore dalla tavolozza attiva; in Alta e Altissima Risoluzione i valori 0 o 2 indicano il nero e i valori 1 o 3 il bianco. Se *paint* è un'espressione stringa l'area viene colorata con una struttura di colori come descritto piú avanti. Se paint è omesso viene assunto il colore di primo piano di default per la grafica.
border	È un'espressione intera compresa tra 0 e 3 che identifica il colore del contorno della figura. In Media Risoluzione *border* specifica il colore in base alla tavolozza

ELEMENTO DI SINTASSI	SIGNIFICATO
	attiva; in Alta e Altissima Risoluzione i valori 0 o 2 rappresentano il nero, i valori 1 o 3 il bianco. Quando inizia la colorazione il contorno deve già avere il colore assegnatogli; la colorazione si interrompe quando viene raggiunto un pixel che ha il colore del contorno. Se *border* è omesso viene assunto per default il valore di *paint*.
background	È un'espressione stringa il cui valore è costituito da un solo carattere, usato per superare la condizione di fine colorazione quando *paint* ha un valore stringa.

Caratteristiche

L'istruzione PAINT colora una figura, avente il contorno del colore specificato da *border*, con il colore specificato da *paint*.

Per esempio, in Media Risoluzione è possibile riempire un cerchio specificando 1 come numero di colore del contorno e 2 come numero di colore di riempimento. Visivamente, questo può significare un cerchio rosso con la circonferenza verde (se è stata selezionata la tavolozza 0).

Poiché in Alta e Altissima Risoluzione ci sono due colori l'istruzione PAINT può essere usata solo per colorare di bianco un'area, finché non viene incontrato il bianco, o viceversa colorare di nero un'area, finché non viene incontrato il nero.

PAINT deve iniziare la colorazione da un punto della figura non appartenente al contorno altrimenti non ha alcun effetto, così pure non ha alcun effetto nel caso in cui il punto specificato è dello stesso colore del contorno della figura.

PAINT può colorare qualsiasi figura, ma la colorazione di figure complesse può causare l'errore "Out of memory", in tal caso occorre usare il comando CLEAR per aumentare la quantità di spazio disponibile per lo stack e rieseguire il programma.

Mosaico

Con l'istruzione PAINT è anche possibile colorare un'area illuminando i suoi pixel con diversi colori, in modo da realizzare una "struttura di colori" (o mosaico).

Per utilizzare questa prestazione occorre impostare, per il parametro *paint*, un valore stringa secondo la seguente sintassi:

PAINT(*x*,*y*),CHR$(*n*) [+ **CHR$(*n*)**]...

---

dove *n* è un numero compreso tra 0 e 255, o tra &H00 e &HFF in esadecimale. È possibile impostare fino a 64 funzioni CHR$(*n*) in ogni istruzione PAINT.

La sequenza delle funzioni stringa CHR$(*n*) permette di generare una "maschera" (tiling patch) a struttura rettangolare con 8 bit di base e con altezza variabile (fino a 64 bit). Questa maschera definisce la struttura di base del mosaico.

La maschera che viene ripetuta in modo uniforme su tutta l'area da riempire e tagliata sul contorno dell'area, è la seguente:

```
     x→
y    x x x x x x x x x   CHR$(n)   byte 0
     x x x x x x x x x   CHR$(n)   byte 1
↓    x x x x x x x x x   CHR$(n)   byte 2
     . . . . . . . . .   . . . .   . . .
     x x x x x x x x x   CHR$(n)   byte 63 (massimo permesso)
```

La ripetizione orizzontale della maschera su video viene fatta in modo tale che ogni riga corrisponde a un byte della memoria video, cioé a 4 pixel in Media Risoluzione e o 8 pixel in Alta e Altissima Risoluzione. Poiché ci sono 80 byte su tutto il video, ci sono 80 possibili posizioni orizzontali della maschera. In verticale, la ripetizione della maschera viene fatta a partire dal margine superiore del video fino al margine inferiore. Ciò significa che una maschera di lunghezza *l* viene ripetuta verticalmente ogni *l* scanline, dove la prima riga corrisponde sempre a una sanline che è un multiplo pari di *l*.

Per conoscere la riga corrispondente ad una data scanline si deve usare la formula:

row = *scanline* **MOD** *l*

dove:

row	identifica la riga della maschera; 0 corrisponde alla prima riga in alto.
scanline	identifica la scanline; 0 corrisponde alla prima scanline.

Usando la seguente istruzione:

PAINT(320,100),CHR$(&H81) + CHR$(&H42) +
CHR$(&H24) + CHR$(&H18) + CHR$(&H18) +
CHR$(&H24) + CHR$(&H42) + CHR$(&H81)

la colorazione inizia dal punto (320, 100) con la riga 4 come prima riga della maschera, poiché *row* = 100 MOD 8.

Con tale istruzione è possibile colorare lo schermo con dei caratteri "x", in Alta e Altissima Risoluzione, infatti la maschera è la seguente:

```
   x →
y  1 0 0 0 0 0 0 1          CHR$ (&H81) byte 0
   0 1 0 0 0 0 1 0          CHR$ (&H42) byte 1
↓  0 0 1 0 0 1 0 0          CHR$ (&H24) byte 2
   0 0 0 1 1 0 0 0          CHR$ (&H18) byte 3
   0 0 0 1 1 0 0 0          CHR$ (&H18) byte 4
   0 0 1 0 0 1 0 0          CHR$ (&H24) byte 5
   0 1 0 0 0 0 1 0          CHR$ (&H42) byte 6
   1 0 0 0 0 0 0 1          CHR$ (&H81) byte 7
```

La stessa istruzione PAINT, in Media Risoluzione, colora lo schermo con una struttura di pixel rossi e verdi (se è attiva la tavolozza 0).

In alcuni casi può essere richiesta la colorazione a mosaico di una figura già colorata; se tale figura ha lo stesso colore specificato da due caratteri consecutivi della maschera la colorazione viene interrotta.

È possibile superare tale limite impostando il parametro *background*, per specificare il colore da non considerare come condizione di fine, assegnandogli un valore che deve coincidere con quello di due caratteri consecutivi della maschera. Supponendo di aver selezionato la tavolozza 0 e di aver impostato CHR$(&HAA), cioè il rosso come colore di *background*, è possibile tracciare, ad esempio, due linee rosse e due blu alternativamente su un colore di fondo rosso.

Se il valore assegnato a *background* coincide con più di due caratteri consecutivi viene emesso il messaggio "Illegal function call"; ciò significa che non è possibile tracciare, ad esempio, più di due linee consecutive rosse su un area già colorata di rosso.

Esempio

```
10 SCREEN 1
20 COLOR 0,0,1,0
30 CLS
40 CIRCLE (256,128),130,2
50 PAINT (256,128),1,2
60 LINE (251,123)-STEP(10,10),0,BF
```

L'istruzione 10 seleziona il modo a Media Risoluzione. L'istruzione 20 seleziona il nero per il numero di colore 0, la tavolozza 0, il verde come colore di primo piano per la grafica, il nero come colore di fondo per la grafica. L'istruzione 30 cancella il video con il colore di fondo (in questo caso il nero). L'istruzione 40 disegna una circonferenza rossa con un raggio di 130 il cui centro è (256,128). L'istruzione 50 colora di verde il cerchio. L'istruzione 60 disegna un rettangolo colorato internamente di nero posto al centro del cerchio.

PEEK Funzione

Restituisce il byte letto dalla locazione di memoria specificata.

PEEK(*offset*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
offset	È un'espressione numerica che restituisce un intero compreso tra 0 e 65535. Indica l'indirizzo della locazione di memoria in cui PEEK legge il byte. Questo indirizzo è dato come spiazzamento rispetto all'origine del segmento corrente, definito dall'ultima istruzione ▶

ELEMENTO DI SINTASSI	SIGNIFICATO
	DEF SEG eseguita. Se non è stata eseguita alcuna istruzione DEF SEG, il segmento corrente è il Segmento Dati GW-BASIC.

Caratteristiche

Il valore restituito è un intero compreso tra 0 e 255.

Se *offset* non è compreso nell'intervallo specificato si verifica l'errore "Illegal function call".

PEEK è la funzione complementare dell'istruzione POKE.

Esempio

A = PEEK(&H5A00)

PLAY Istruzione

Produce musica in base a una stringa che specifica le note, che devono essere suonate, e il modo in cui devono essere suonate.

PLAY *stringexp*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
stringexp	È un'espressione stringa che contiene una serie di comandi di un solo carattere.

Caratteristiche

PLAY usa un concetto simile a quello di DRAW; cioé prevede l'inclusione di un Music Macro Language, in un'unica istruzione. Una serie di sottocomandi, usati nell'istruzione PLAY, specifica quale azione particolare deve essere intrapresa.

I sottocomandi usati per *stringexp* sono:

COMANDO	AZIONE
A-G[#| + | -]	Suona una nota compresa nell'intervallo A-G. I suffissi (#) o (+) dopo la nota specificano il diesis, il suffisso (-) specifica il bemolle.
O*n*	Imposta l'ottava corrente. Ci sono sette ottave, numerate da 0 a 6.
>*n*	Incrementa l'ottava e suona la nota *n*. L'ottava viene progressivamente incrementata, ogni volta che la nota *n* viene

COMANDO	AZIONE
	suonata, finché non viene raggiunta l'ottava 6. La nota *n* viene in seguito suonata all'ottava 6.
<*n*	Diminuisce l'ottava e suona la nota *n*. L'ottava viene progressivamente diminuita, ogni volta che la nota *n* viene suonata, finché non viene raggiunta l'ottava 0. La nota *n* viene in seguito suonata all'ottava 0.
N*n*	Suona una delle 84 note comprese tra le 7 possibili ottave. Il parametro *n* varia da 0 a 84; 0 indica una pausa. Questo comando è un altro modo per specificare le note usando il nome della nota (A-G) ed i comandi del numero di ottava.
P*n*	Specifica una pausa. Il parametro *n* varia da 1 a 64 e corrisponde alla lunghezza di ogni nota, impostata da L*n*.
L*n*	Imposta la lunghezza di ogni nota. Il parametro *n* varia da 1 a 64, dove *n* = 1 equivale ad una nota intera; *n* = 4 equivale ad un quarto di nota, ecc. La lunghezza può anche seguire la nota quando per una nota particolare si richiede solo un cambio di lunghezza. In questo caso A16 e quivale a L16A.

COMANDO	AZIONE
.	Un punto dopo una nota fa suonare la nota per 3/2 volte la lunghezza, determinata da L moltiplicata per T. Una nota può essere seguita da più punti. La durata della nota è proporzionata di conseguenza; p.e.: A. è 3/2, A.. è 9/4, A... è 27/8, ecc. Anche una pausa (P) può essere seguita da punto. Anche in questo caso la lunghezza della pausa viene calcolata nello stesso modo delle note.
T*n*	Imposta il tempo, o numero di quarti, in un minuto. Il parametro *n* varia da 32 a 255, con un valore di default pari a 120.
MF	Imposta il Foreground della Musica. Le istruzioni PLAY e SOUND sono eseguite in foreground. Ogni nota o suono successivo non incomincia finché la nota o il suono precedente non è finito. Questa è la procedura di default.
MB	Imposta il Background della Musica. Le istruzioni PLAY e SOUND sono eseguite in background. Ogni nota o suono sono memorizzati in un buffer che permette al programma GW-BASIC di continuare l'esecuzione mentre le note o i suoni vengono eseguiti in background. Nel background possono essere suonate fino a 32 note contemporaneamente.

COMANDO	AZIONE
MN	Imposta il "modo normale" in cui ogni nota dura 7/8 del tempo stabilito dalla lunghezza (L).
ML	Imposta il "modo legato" in cui ogni nota dura per tutto il tempo stabilito dalla lunghezza (L).
MS	Imposta il "modo staccato" in cui ogni nota dura 3/4 del tempo stabilito dalla lunghezza (L).
string x	Esegue la stringa specificata.

Osservazioni

Il parametro *n* può essere costante o variabile, dove una variabile è scritta = *variable*;. Il punto e virgola è necessario quando una variabile viene usata in questo modo, o quando viene usato il comando X, ma non è ammesso dopo MF, MB, MN, ML o MS. In tutti gli altri casi, un punto e virgola tra i comandi è facoltativo.

Esempio 1

```
10 LET LISTEN$ = "T180 O2 P2 P8 L8 GGG L2 E-"
20 LET FATE$ = "P24 P8 L8 FFF L2 D"
30 PLAY LISTEN$ + FATE$
```

Questo esempio suona l'inizio del primo movimento della Quinta Sinfonia di Beethoven.

Esempio 2

```
100 PLAY "<<" 'Diminuisce di due ottave
200 PLAY ">" 'Aumenta di un'ottava
300 PLAY "A>" 'Aumenta di un'ottava e suona la nota A
400 PLAY "X SONG$"
```

PLAY(*n*) Funzione

Restituisce il numero di note presenti nel buffer di background della musica.

PLAY(*n*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È un argomento fittizio; può essere fornito un qualsiasi valore.

Caratteristiche

SE...	ALLORA...
il programma viene eseguito in modo Music Foreground	PLAY(*n*) restituisce 0.

▶

SE...	ALLORA...
il programma viene eseguito in modo Music Background	PLAY(*n*) restituisce il numero di note presenti nel buffer di background della musica. Il valore massimo restituito da PLAY(*n*) è 32.

Esempio

```
10 IF PLAY(0) = 6 GOTO 500
```

PLAY {ON I OFF I STOP} Istruzione

PLAY ON abilita il rilevamento PLAY(*n*).

PLAY OFF disabilita il rilevamento PLAY(*n*).

PLAY STOP sospende il rilevamento PLAY(*n*).

PLAY{ON I OFF I STOP}

Caratteristiche

PLAY ON, PLAY OFF, PLAY STOP vengono usate insieme all'istruzione ON PLAY(*n*) GOSUB.

Se è stata eseguita un'istruzione PLAY OFF, l'istruzione GOSUB non viene eseguita e non è memorizzata.

Se è stata eseguita un'istruzione PLAY STOP, l'istruzione GOSUB non viene eseguita mentre viene eseguita subito dopo l'istruzione PLAY ON.

Se viene eseguita ON PLAY(*n*) GOSUB, viene eseguita PLAY STOP automaticamente.

Quando ha avuto luogo un rilevamento di errore, tutte le routine di rilevamento vengono automaticamente disabilitate.

L'istruzione RETURN *linenum* può essere usata per ritornare dal sottoprogramma di rilevamento ad un numero di linea specificato. Si usi comunque questo tipo di ritorno con attenzione, poiché qualsiasi altra istruzione GOSUB, WHILE o FOR che era attiva al momento del rilevamento rimane attiva e si posssono generare errori come "FOR without NEXT".

PMAP Funzione

Trasforma le coordinate fisiche in coordinate logiche e viceversa (solo modo grafica).

PMAP(*coordinate,n*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
coordinate	È un'espressione numerica che specifica la coordinata *x* o la coordinata *y* del punto in relazione al valore di *n*.
n	È un numero intero compreso tra 0 e 3. 0 specifica che il valore di *coordinate* è la coordinata logica *x* e che PMAP restituisce il valore della coordinata fisica *x*. 1 specifica che il valore di *coordinate* è la coordinata logica *y* e che PMAP restituisce il valore della coordinata fisica *y*. 2 specifica che il valore di *coordinate* è la coordinata fisica *x* e che PMAP restituisce il valore della coordinata logica *x*. 3 specifica che il valore di *coordinate* è la coordinata fisica *y* e che PMAP restituisce il valore della coordinata logica *y*.

Caratteristiche

Le quattro funzioni PMAP permettono di trovare le posizioni equivalenti dei punti sulla finestra logica, definita dall'istruzione WINDOW, e di quelli sulla finestra fisica, definita dall'istruzione VIEW.

Esempi

Se l'utente ha definito WINDOW SCREEN (80,100)-(200,200), il vertice superiore di sinistra della finestra logica è (80,100) e quello inferiore di destra è (200,200). Con coordinata logica 2 le coordinate fisiche sono (0,0) nel vertice superiore di sinistra e (639,199) in quello inferiore di destra, allora:

X = PMAP(80,0)

calcola la coordinata fisica *x* corrispondente alla coordinata logica *x*, pari a 80:

0

La funzione PMAP nella seguente istruzione:

Y = PMAP (200,1)

calcola la coordinata fisica *y* corrispondente alla coordinata logica *y*, pari a 200:

199

La funzione PMAP nella seguente istruzione:

X = PMAP (639,2)

calcola la coordinata logica *x* corrispondente alla coordinata fisica *x*, pari a 639:

200

La funzione PMAP nella seguente istruzione:

Y = PMAP(199,3)

calcola la coordinata logica *y* corrispondente alla coordinata fisica *y*, pari a 199:

200

POINT Funzione

Restituisce il numero di colore di un pixel su video, se vengono specificati due argomenti (*x*,*y*). Retituisce la coordinata *x* o *y* dell'ultimo punto riferito, se viene specificato un solo parametro (*n*).

Sintassi 1

POINT(*x*,*y*)

Sintassi 2

POINT(*n*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
x,*y*	Sono le coordinate fisiche assolute del pixel selezionato. Se il punto è fuori dalla finestra fisica corrente, viene restituito il valore -1.
n	*n* può assumere i valori:

ELEMENTO DI SINTASSI	SIGNIFICATO
	0 specifica la coordinata fisica *x* corrente. 1 specifica la coordinata fisica *y* corrente. 2 specifica la coordinata universale logica *x* corrente, se è stata eseguita una istruzione WINDOW, altrimenti viene restituito lo stesso valore di POINT(0). 3 specifica la coordinata logica *y* corrente, se è stata eseguita una istruzione WINDOW altrimenti viene restituito lo stesso valore di POINT(1).

Caratteristiche

v% = POINT(*x*,*y*)

restituisce il numero di colore del pixel specificato e lo memorizza nella variabile intera *v%*.

In Media Risoluzione, POINT(*x*,*y*) può restituisce 0, 1, 2 o 3; in Alta o Altissima Risoluzione può restituire 0 o 1.

v2 = POINT(1)

restituisce la coordinata fisica *y* del punto corrente e la memorizza nella variabile in semplice (o doppia) precisione *v2*.

Esempio

```
10 SCREEN 2
20 IF POINT (I,I) < >0 THEN PRESET (I,I) ELSE PSET (I,I)
30 'Inverte lo stato corrente del punto (I,I)
40 PSET (I,I), 1 - POINT (I,I)
50 'Un altro modo per invertire il punto, se il sistema è B/N.
```

Esempio

```
10 SCREEN 1
20 LET C=3
30 PSET (10,10),C
40 IF POINT (10,10)=C THEN PRINT "This point is color"; C
```

POKE Istruzione

Scrive un byte in una locazione di memoria.

POKE *offset* , *byte*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
offset	È un'espressione numerica che restituisce un intero compreso tra 0 e 65535 e indica l'indirizzo della locazione di memoria in cui l'istruzione POKE scrive un byte. L'indirizzo viene espresso come spiazzamento rispetto all'origine del segmento corrente, specificato dall'ultima istruzione DEF SEG eseguita. Se non è stata eseguita alcuna istruzione DEF SEG, il segmento corrente è il Segmento Dati GW-BASIC.
byte	Rappresenta il byte che l'istruzione scrive all'indirizzo specificato. Deve essere compreso tra 0 e 255.

Caratteristiche

L'istruzione POKE e la funzione PEEK vengono utilizzate per il trasferimento di argomenti e risultati da sottoprogrammi in linguaggio macchina.

Se *offset* o *byte* si trovano fuori dall'intervallo specificato, si verifica l'errore "Illegal function call".

PEEK è la funzione complementare di POKE.

Attenzione

L'istruzione POKE deve essere usata con attenzione, per evitare eventuali interruzioni di GW-BASIC o MS-DOS.

Esempio

```
10 POKE &H5A00,&HFF
```

POS Funzione

Restituisce la posizione orizzontale del cursore.

POS(*dummy*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
dummy	È un argomento fittizio. Viene accettato qualsiasi valore.

Caratteristiche

La funzione POS restituisce la posizione orizzontale (numero di colonna) del cursore. La posizione più a sinistra è 1, mentre quella più a destra può essere 40 o 80, a seconda dell'ampiezza del video. Per trovare la posizione verticale (numero di riga) del cursore è possibile usare la funzione CSRLIN.

Esempio

```
IF POS(0)>70 THEN BEEP
```

PRESET Istruzione

Cancella o illumina un punto su video in una posizione specificata (solo modo grafica).

PRESET [STEP] (*x*,*y*) [,*color*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
x,y	Sono le coordinate del punto e possono essere in forma assoluta o relativa (se viene inclusa l'opzione STEP).
color	È un'espressione intera il cui valore è compreso tra 0 e 3 e rappresenta il numero di colore del punto specificato. In Media Risoluzione *color* sceglie il colore dalla tavolozza attiva; in Alta e Altissima Risoluzione i valori 0 o 2 indicano il nero e i valori 1 o 3 il bianco. Se *color* è omesso, viene selezionato il colore di fondo per la grafica. Se *color* viene specificato, l'istruzione PRESET ha la stessa funzione dell'istruzione PSET

Esempio

PRESET (*x*,*y*)

cancella dal video il punto alle coordinate *x* e *y*; ha la stessa funzione di:

PSET (*x*,*y*),0

se il colore di fondo per la grafica è 0.

Errori Possibili

Se viene specificato un colore maggiore di 3 si verifica l'errore "Illegal function call".

PRINT Istruzione

Visualizza un insieme di dati.

PRINT [*list-of-expressions*[, | ;]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
list-of-expressions	Le espressioni della lista possono essere di tipo numerico e/o stringa (le costanti stringa devono essere racchiuse

ELEMENTO DI SINTASSI	SIGNIFICATO
	tra doppi apici). Ogni espressione deve essere separata dalla seguente da una virgola, da uno spazio o da un punto e virgola.

Caratteristiche

Se viene omesso il parametro *list-of-expressions*, viene visualizzata una linea di spazi. Se invece *list-of-espressions* viene incluso, i valori delle espressioni vengono visualizzati sul terminale.

La posizione di ogni dato visualizzato viene determinato dalla punteggiatura usata per separare gli elementi della lista. GW-BASIC divide la linea in zone di visualizzazione di 14 spazi ognuna. Nella lista di espressioni, il valore che segue una virgola viene visualizzato all'inizio della zona successiva; il valore che segue un punto e virgola, viene visualizzato immediatamente dopo l'ultimo valore. Uno o più spazi tra le espressioni provocano lo stesso effetto di un punto e virgola.

Se l'ultima espressione della lista è seguita da una virgola o da un punto e virgola, l'istruzione PRINT o PRINT USING successiva inizia a visualizzare i dati sulla stessa linea, se c'è spazio sufficiente. Se la lista di espressioni non è seguita da una virgola o da un punto e virgola, viene impostato **CR** alla fine della linea. Se la linea visualizzata supera in lunghezza l'ampiezza del terminale, GW-BASIC si sposta alla successiva linea fisica e continua la visualizzazione.

I numeri visualizzati vengono sempre seguiti da uno spazio. I numeri positivi sono preceduti da uno spazio, mentre quelli negativi sono preceduti dal segno meno. I numeri in semplice precisione

che possono essere rappresentati con 7 o meno di 7 cifre, vengono visualizzati in formato a virgola fissa. Per esempio, 1E-7 viene visualizzato come .0000001 e 1E-8 viene visualizzato come 1E-08. I numeri in doppia precisione che possono essere rappresentati con 16 oppure meno di 16 cifre in formato a virgola fissa, con la stessa accuratezza del formato a virgola mobile, vengono visualizzati in formato a virgola fissa. Per esempio: 1D-15 viene visualizzato come .0000000000000001 e 1D-16 viene visualizzato come 1D-16.

In un'istruzione PRINT è possibile usare un punto interrogativo al posto della parola PRINT.

Esempio 1

```
10 X=5
20 PRINT X+5,X-5,X*(-5),X^5
30 END
RUN
 10     0     -25     3125
Ok
```

In questo esempio, le virgole presenti nell'istruzione PRINT provocano la visualizzazione di ogni valore all'inizio della zona successiva.

Esempio 2

```
LIST
10 INPUT X
20 PRINT X "SQUARED IS" X^2"AND";
30 PRINT X "CUBED IS" X^3
40 PRINT
50 GOTO 10
Ok
RUN
? 9
 9 SQUARED IS 81 AND 9 CUBED IS 729
```

```
? 21
 21 SQUARED IS 441 AND 21 CUBED IS 9261

?
```

In questo esempio, il punto e virgola alla fine della linea 20 provoca la visualizzazione sulla stessa linea di entrambe le istruzioni PRINT. La linea 40 provoca la visualizzazione di una linea di spazi prima dei prompt.

Esempio 3

```
10 FOR X=1 TO 5
20 J=J+5
30 K=K+10
40 ? J;K;
50 NEXT X
Ok
RUN
 5  10  10  20  15  30  20  40  25  50
Ok
```

In questo esempio, il punto e virgola presente nella istruzione PRINT provoca la visualizzazione di ogni valore immediatamente dopo il valore precedente (un numero è sempre seguito da uno spazio e i numeri positivi sono preceduti da uno spazio). Nella linea 40 viene usato un punto interrogativo al posto della parola PRINT.

PRINT USING Istruzione

Visualizza un insieme di dati in un formato specificato.

PRINT USING *format-string* ; *list-of-expressions*[, | ;]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
format-string	È un'espressione stringa (di solito una costante o una variabile) composta da speciali caratteri di formattazione, che determinano il campo e il formato delle stringhe o dei numeri visualizzati.
list-of-expressions	È composta dalle espressioni numeriche o stringa da visualizzare, separate da punti e virgola o virgole. Le costanti stringa devono essere racchiuse tra doppi apici. Se la lista di espressioni termina con una virgola o un punto e virgola, l'istruzione PRINT o PRINT USING successiva inizia a stampare sulla stessa linea, se c'è spazio a sufficienza.

Campi Stringa

Se per visualizzare stringhe viene usata l'istruzione PRINT USING, per formattare il campo stringa è possibile usare uno dei tre seguenti caratteri di formattazione:

CARATTERE DI FORMATTAZIONE	SIGNIFICATO
!	Specifica che solo il primo carattere della stringa data deve essere visualizzato.

CARATTERE DI FORMATTAZIONE	SINGNIFICATO
n spazi\\	Specifica che devono essere visualizzati 2 + *n* caratteri della stringa. Se le barre inverse non contengono spazi, vengono visualizzati due caratteri; se contengono uno spazio, vengono visualizzati tre caratteri, e così via. Se la stringa è più lunga del campo, i caratteri eccedenti vengono ignorati. Se invece il campo è più lungo della stringa, la stringa viene allineata alla sinistra del campo e viene completata con spazi sulla destra. Per esempio 10 A$ = "LOOK":B$ = "OUT" 30 PRINT USING "!";A$;B$ 40 PRINT USING"\\ \\";A$;B$ 50 PRINT USING"\\ \\";A$;B$;"!!" **RUN** LO LOOKOUT LOOK OUT !!
&	Specifica un campo stringa di lunghezza variabile. Quando il campo viene specificato con "&", la stringa viene visualizzata esattamente come è stata introdotta.

CARATTERE DI FORMATTAZIONE	SIGNIFICATO
	Per esempio: 10 A$ = "LOOK":B$ = "OUT" 20 PRINT USING "!";A$; 30 PRINT USING "&";B$ **RUN** LOUT

Campi Numerici

Quando viene usata l'istruzione PRINT USING per visualizzare numeri, per formattare il campo numerico è possibile usare i caratteri speciali di formattazione:

CARATTERE DI FORMATTAZIONE	SIGNIFICATO
#	Rappresenta la posizione di ciascuna cifra. Le posizioni delle cifre sono sempre riempite. Se il numero da visualizzare è inferiore al numero delle posizioni specificate, il numero viene allineato a destra (preceduto da spazi) nel campo.
	È possibile inserire un punto decimale in qualunque posizione del campo. Se la stringa di formattazione indica che una cifra deve precedere il punto decimale, la cifra viene sempre visualizzata

CARATTERE DI FORMATTAZIONE	SIGNIFICATO
	(come 0 se necessario). Quando è necessario i numeri sono arrotondati. **PRINT USING "# #.# #";.78** 0.78 **PRINT USING "# # #.# #";987.654** 987.65 **PRINT USING "# #.# # ";10.2,5.3,66.789,.234** 10.20 5.30 66.79 0.23 Nell'ultimo esempio, alla fine della stringa di formattazione è stato inserito uno spazio per separare i valori visualizzati sulla linea.
+	Un segno più all'inizio o alla fine della stringa di formattazione provoca la visualizzazione del segno (più o meno) prima o dopo il numero.
-	Un segno meno alla fine del campo di formattazione provoca la visualizzazione di numeri negativi seguiti dal segno meno. **PRINT USING"+ # #.# # ".68.95,2.4,55.6,-.9** -68.95 +2.40 +55.60 -0.90

▶

CARATTERE DI FORMATTAZIONE	SIGNIFICATO
	PRINT USING"# #.# #- ";-68.95,22.449,-7.01 68.95- 22.45 7.01-
**	Un doppio asterisco all'inizio della stringa di formattazione provoca il riempimento degli spazi iniziali di un campo numerico con asterischi. Il simbolo ** riserva posizioni per due ulteriori cifre. **PRINT USING "**#.# ";12.39,-0.9,765.1** *12.4 *-0.9 765.1
$$	Un doppio segno dollaro provoca la visualizzazione di un segno dollaro alla sinista del numero formattato. Il segno $$ specifica due ulteriori posizioni di cifra, una delle quali è occupata dal segno dollaro. Il carattere $$ non può essere usato con il formato esponenziale. I numeri negativi non possono venire usati, se l'ultimo carattere della stringa di formato non è un segno meno. **PRINT USING "$$# # #.# #";456.78** $456.78
**$	Tale segno situato all'inizio di una stringa di formattazione combina gli effetti dei due simboli appena descritti. Gli spazi iniziali vengono riempiti da

CARATTERE DI FORMATTAZIONE	SIGNIFICATO
	asterischi ed un segno dollaro viene inserito immediatamente prima del numero. **$ specifica tre ulteriori posizioni di cifra, una delle quali è il segno dollaro. **PRINT USING"**$##.##";2.34** ***$2.34
,	Una virgola alla sinistra del punto decimale in una stringa di formattazione provoca la visualizzazione di una virgola alla sinistra di ogni terza cifra, nell'ambito della parte intera del numero. Se una virgola è presente alla fine della stringa di formattazione questa viene visualizzata come parte della stringa stessa. Questo carattere specifica un'altra posizione di cifra e non esercita alcun effetto se usato con un formato esponenziale (∧∧∧∧). **PRINT USING "####,.##";1234.5** 1,234.50 **PRINT USING "####.##,";1234.5** 1234.50,
∧∧∧∧	Il formato esponenziale può essere specificato inserendo quattro simboli di elevamento a potenza (o frecce in alto) dopo i caratteri che specificano la posizione delle cifre. Tali simboli permettono di generare lo spazio per la visualizzazione di E + *xx* o D + *xx*. È possibile specificare qualsiasi posizione del punto decimale. Le cifre significative sono allineate a sinistra e l'esponente viene posizionato

CARATTERE DI FORMATTAZIONE	SIGNIFICATO
	di conseguenza. Se non viene specificato un segno + iniziale o finale, oppure un segno -, viene utilizzata una posizione di cifra alla sinistra del punto decimale per visualizzare uno spazio oppure un segno meno. **PRINT USING "##.## ^^^^";234.56** 2.35E+02 **PRINT USING ".#### ^^^^-";888888** .8889E+06 **PRINT USING "+.## ^^^^";123** +.12E+03
_	Un segno di sottolineatura incluso nella stringa di formattazione provoca la visualizzazione del successivo carattere visualizzato così come compare nella stringa di formattazione. **PRINT USING "_!##.##_!";12.34** !12.34! Il carattere visualizzato così come compare nella stringa di formattazione può essere lo stesso segno di sottolineatura: in questo caso si include "_ _" nella stringa di formattazione.
%	Se il numero che deve essere visualizzato è maggiore del campo numerico specificato, davanti

CARATTERE DI FORMATTAZIONE	SIGNIFICATO
	al numero compare il segno di percentuale. Se per effetto dell'arrotondamento il numero supera il campo, anche in tal caso davanti ad esso compare il segno di percentuale. **PRINT USING "# #.# #";111.22** %111.22 **PRINT USING ".# #";.999** %1.00 Se il numero di cifre specificate è superiore a 24, si verifica l'errore "Illegal function call".

PRINT# e PRINT# USING Istruzioni

Scrivono dati in sequenza su un file su disco.

Le istruzioni PRINT# e PRINT# USING vengono generalmente usate all'interno di un programma.

PRINT # *filenum*,[**USING** *format-string*;]*list-of-expressions*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filenum	È il numero con cui il file è stato aperto con l'opzione OUTPUT.
format-string	È un'espressione stringa (di solito una costante o una variabile) composta da caratteri di formattazione.
list-of-expressions	È una lista di espressioni di tipo numerico e/o stringa da scrivere sul file.

Caratteristiche

L'istruzione PRINT# non compatta i dati su disco. Poiché i dati vengono registrati su disco nello stesso formato in cui sono visualizzati con l'istruzione PRINT, è necessario registrare i dati con gli opportuni delimitatori, in modo da poterli poi rileggere in modo corretto dal disco.

Nella lista di espressioni, le espressioni numeriche devono essere delimitate da punti e virgola. Per esempio:

```
50 PRINT#1,B;C;D;X;Y;Z
```

Se come delimitatori vengono usate le virgole, gli spazi addizionali inseriti tra i campi da visualizzare vengono anche scritti sul disco.

Le espressioni stringa presenti nella lista devono essere separate da punti e virgola. Per formattare correttamente le espressioni stringa sul disco, si devono usare i delimitatori espliciti nella lista di espressioni.

Per esempio, A$="CAMERA" e B$="93604-1". L'istruzione

```
100 PRINT#1,A$;B$
```

scriverà sul disco CAMERA93604-1. Poiché non ci sono delimitatori, questa stringa non può essere letta da disco come due stringhe separate. Per poter fare ciò, bisogna inserire delimitatori espliciti nell'istruzione PRINT# come segue:

```
200 PRINT#1,A$;",";B$
```

L'immagine scritta sul disco è allora:

```
CAMERA,93604-1
```

che può venire letta come due variabili stringa. Se le stringhe stesse contengono virgole, punti e virgola, spazi iniziali significativi, ritorni a capo o interlinee, occorre scriverle su disco racchiuse fra doppi apici usando CHR$(34). Per esempio, sia A$="CAMERA, AUTOMATIC" e B$=" 93604-1".

L'istruzione

```
300 PRINT#1,A$;B$
```

scrive sul disco la seguente stringa:

```
CAMERA, AUTOMATIC 93604-1
```

L'istruzione

```
400 INPUT#1,A$,B$
```

assegna la stringa "CAMERA" alla variabile A$ e la stringa "AUTOMATIC 93604-1" a B$. Per separare correttamente questa stringa sul disco, bisogna inserire i doppi apici nella stringa registrata sulla copia del disco usando la funzione CHR$(34). L'istruzione

```
500 PRINT#1,CHR$(34);A$;CHR$(34);CHR$(34);B$;CHR$(34)
```

registra su disco la seguente stringa:

```
"CAMERA, AUTOMATIC" "93604-1"
```

Invece, l'istruzione

```
600 INPUT#1,A$,B$
```

assegna la stringa "CAMERA, AUTOMATIC" alla variabile A$ e la stringa "93604-1" a B$. L'istruzione PRINT# può anche essere usata con l'opzione USING per controllare il formato del file su disco. Per esempio:

```
700 PRINT#1,USING"$$###.##,";J;K;L
```

Per ulteriori esempi sull'uso di PRINT#, occorre fare riferimento al Capitolo 4 e all'istruzione WRITE#.

PSET Istruzione

Illumina un punto su video con un dato colore (solo modo grafica).

PSET [STEP] (*x*,*y*) [,*color*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
x,y	Sono le coordinate del pixel da illuminare e possono essere espresse in forma assoluta o relativa.
color	È un'espressione intera il cui valore è compreso tra 0 e 3 e rappresenta il numero di colore del punto specificato. In Media Risoluzione *color* sceglie il colore dalla tavolozza attiva; in Alta e Altissima Risoluzione i valori 0 o 2 rappresentano il nero e i valori 1 o 3 il bianco. Se *color* è omesso viene selezionato il colore di primo piano per la grafica.

Caratteristiche

Le coordinate possono essere specificate in una delle due forme:

PSET STEP (*x-offset*, *y-offset*) oppure
PSET (*absolute-x*, *absolute-y*)

La prima forma seleziona un punto facendo riferimento al punto corrente. La seconda forma seleziona un punto specificando le sue coordinate assolute facendo riferimento al sistema di coordinate in vigore.

Esempio 1

PSET(10,10) forma assoluta
PSET STEP(10,0) offset 10 in x e 0 in y
PSET(0,0) origine

Esempio 2

In questo esempio si traccia un segmento diretto dal punto (0,0) al punto (100,100):

```
10 FOR I=0 TO 100
20 PSET (I,I)
30 NEXT
```

quindi viene cancellato il segmento diretto rendendo invisibile ogni suo punto:

```
40 FOR I=100 TO 0 STEP -1
50 PSET (I,I),0
60 NEXT
```

PUT (File) Istruzione

Scrive un recod su un file ad accesso diretto prendendolo da un buffer ad accesso diretto.

L'istruzione PUT (File) viene generalmente usata all'interno di un programma.

PUT [#]*filenum*[,*recordnum*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filenum	È il numero con cui il file è stato aperto.
recordnum	È il numero del record da scrivere, compreso tra 1 e 16777215. Se omesso, viene assunto il numero di record corrente (cioé, il record il cui numero è superiore di 1 all'ultimo record trattato).

Caratteristiche

Le istruzioni PRINT#, PRINT# USING, WRITE#, LSET e RSET possono essere usate per introdurre caratteri nel buffer del file ad accesso diretto prima di eseguire un'istruzione PUT.

Se viene usata l'istruzione WRITE#, GW-BASIC riempie il buffer di spazi fino al ritorno a capo.

Esempio

```
LIST
10 OPEN "R",1,"A:RAND",48
20 FIELD 1,20 AS R1$,20 AS R2$,8 AS R3$
30 FOR L=1 TO 4
40 INPUT "name";N$
50 INPUT "address";M$
60 INPUT "phone";P$
70 LSET R1$=N$
80 LSET R2$=M$
90 LSET R3$=P$
100 PUT 1,L
110 NEXT L
```

```
120 CLOSE 1
130 END
Ok
RUN
name? super man
address?USA
phone? 11234621
name? robin hood
address? England
phone? 23462101
.
.
.
Ok
```

L'istruzione 10 apre il file ad accesso diretto RAND, con una lunghezza di record di 48 byte, sul diskette inserito nel drive A. Il numero del file è 1. L'istruzione 20 divide il buffer in campi.

L'istruzione 100 registra un record sul file RAND, con il numero di record uguale al valore della variabile di controllo del ciclo FOR/NEXT.

Errori Possibili

Se si tenta di leggere o scrivere oltre la fine del buffer, si verifica l'errore "Field overflow".

PUT (File COM) Istruzione

Scrive un numero specifico di byte su un file di comunicazione.

L'istruzione PUT (File COM) viene generalmente usata all'interno di un programma.

PUT [#]*filenum*[,*length*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filenum	È un'espressione intera il cui valore è un numero di file.
length	È un'espressione intera il cui valore è il numero di byte da trasferire dal buffer di comunicazione; *length* non può superare il valore fissato dalla clausola LEN nell'istruzione OPEN COM.

Esempio

```
100 PUT #2,80
```

PUT (Grafica) Istruzione

Visualizza l'immagine memorizzata in un vettore tramite l'istruzione GET.

PUT (*x*,*y*) ,*array* [,*action-verb*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
x,y	Rappresentano le coordinate del vertice in alto a sinistra del rettangolo da visualizzare.
array	È il nome di un vettore contenente l'immagine da visualizzare. Il vettore deve essere di tipo numerico.
action-verb	È una stringa di caratteri che specifica un'operazione logica tra i colori dei punti da visualizzare e i colori dei punti presenti su video. La stringa può assumere i valori: PSET, PRESET, AND, OR, XOR. Per default viene assunto XOR.

Caratteristiche

L'istruzione PUT viene usata per trasferire immagini su video prelevandole da un vettore in memoria (dove erano state memorizzate con GET); permette inoltre di realizzare l'animazione e il movimento di oggetti.

Il vettore, che può essere di un qualsiasi tipo numerico, contiene l'immagine e deve essere stato dimensionato in modo da memorizzare i colori di tutti i pixel dell'immagine stessa.

Il Parametro *action-verb*

Il parametro *action-verb* viene usato per realizzare operazioni logiche tra i colori dei punti da visualizzare e i colori dei pixel presenti su video.

Può assumere i seguenti valori:

- PSET, visualizza i dati pixel per pixel. Ogni pixel è dello stesso colore di quando è stato trasferito dal video in memoria.

- PRESET, svolge la stessa azione di PSET, ma produce un'immagine in negativo.

- AND, viene usato per trasferire l'immagine solo se esiste già un'immagine visualizzata sul video. I punti che sono dello stesso colore, sia nell'immagine esistente che nell'immagine da visualizzare, rimangono dello stesso colore, mentre i punti che hanno colore diverso sono trasformati eseguendo l'operazione AND tra i corrispondenti bit.

- OR, viene usato per sovrapporre l'immagine su quella già esistente.

- XOR, viene usato per visualizzare oggetti in movimento. XOR fa si che un punto su video venga invertito se ad esso corrisponde un punto nell'immagine memorizzata nel vettore. Quando viene visualizzata un'immagine, su un colore di fondo complesso, per due volte nella stessa posizione, l'immagine viene cancellata ma il colore di fondo non subisce modifiche. Questo permette di visualizzare un oggetto in movimento senza alterare il colore di fondo.

In Media Risoluzione AND, OR e XOR influenzano il colore in questo modo:

AND

vettore \ video	0	1	2	3
0	0	0	0	0
1	0	1	0	1
2	0	0	2	2
3	0	1	2	3

OR

vettore \ video	0	1	2	3
0	0	1	2	3
1	1	1	3	3
2	2	3	2	3
3	3	3	3	3

XOR

vettore \ video	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

Oggetti in Movimento

Le istruzioni GET e PUT consentono di visualizzare oggetti in movimento, con la seguente procedura:

1. Visualizzare l'immagine (tramite XOR).

2. Calcolare la nuova posizione del rettangolo che contiene l'immagine.

3. Visualizzare l'immagine (tramite XOR) una seconda volta sulla posizione precedente, per cancellare l'immagine precedente.

4. Tornare al punto 1, visualizzando questa volta l'immagine nella nuova posizione.

In questo caso il colore di fondo non viene modificato. È possibile ridurre lo sfarfallio riducendo al massimo i tempi tra il punto 4 ed il punto 1 e assicurandosi (eventualmente con un ciclo di ritardo FOR/NEXT) che vi sia un intervallo di tempo sufficiente tra 1 e 3. Se vengono visualizzati più oggetti in movimento, il procedimento deve essere ripetuto per ogni singolo oggetto.

Se non è necessario conservare il colore di fondo, è possibile visualizzare gli oggetti in movimento tramite *action-verb* PSET. Questo metodo consiste nel lasciare intorno all'immagine, quando viene visualizzata per la prima volta, un margine uguale o maggiore della distanza massima che dovrà percorrere l'oggetto. Perciò quando l'oggetto si sposta questo margine cancella effettivamente l'immagine precedente. Questo metodo può essere più veloce di quello precedentemente descritto (tramite XOR), poiché è necessaria una sola istruzione PUT per far muovere un oggetto (anche se è necessario visualizzare un'immagine più grande).

Errori Possibili

Se l'immagine da visualizzare supera i limiti del video, si verifica l'errore "Illegal function call".

RANDOMIZE Istruzione

Modifica il generatore di numeri casuali.

RANDOMIZE [*numexp*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
numexp	È una qualsiasi espressione intera che viene utilizzata per definire il punto iniziale di una nuova sequenza di numeri casuali.

Caratteristiche

Se viene omesso il parametro *numexp*, GW-BASIC sospende l'esecuzione del programma e richiede un nuovo valore visualizzando:

Random Number Seed (-32768 to 32767)?

prima di eseguire l'istruzione RANDOMIZE.

Per definire un nuovo punto iniziale di numeri casuali senza prompt si deve usare la funzione numerica TIMER. Per esempio:

RANDOMIZE TIMER

Se il generatore dei numeri casuali non viene reinizializzato, la funzione RND fornisce la stessa sequenza di numeri casuali ogni volta che il programma viene eseguito. Per modificare la sequenza dei numeri casuali per ogni esecuzione del programma, si può porre l'istruzione RANDOMIZE all'interno del programma e poi modificare l'argomento ad ogni esecuzione.

Esempio

```
10 RANDOMIZE
20 FOR I=1 TO 5
30 PRINT RND;
40 NEXT I
RUN
Random Number Seed (-32768 to 32767)? 3
 .2226007  .5941419  .2414202  .2013798  5.361748E-02
Ok
RUN
Random Number Seed (-32768 to 32767)? 4
 .628988  .765605  .5551561  .775797  .7834911
Ok
RUN
Random Number Seed (-32768 to 32767)? 3
 .2226007  .5941419  .2414202  .2013798  5.361748E-02
Ok
```

READ Istruzione

Legge i valori da una o più istruzioni DATA e li assegna alle variabili specificate. L'istruzione READ viene generalmente usata all'interno di un programma.

READ *variable*[,*variable*]...

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
variable	Ogni variabile dell'elenco può essere di tipo numerico o stringa. Il tipo della variabile deve essere coerente con il tipo del valore che le viene associato nella sequenza delle istruzioni DATA.

Caratteristiche

Se un programma contiene un'istruzione READ deve anche contenere almeno un'istruzione DATA. L'istruzione READ assegna alle variabili specificate i valori indicati da DATA, in particolare alla prima variabile della lista viene assegnato il primo valore della lista, e così via. Se il tipo di dato (numerico o stringa) non corrisponde al tipo della variabile associata, si verifica l'errore "Syntax error". Comunque è possibile assegnare qualsiasi tipo di dato numerico (intero, in semplice o doppia precisione) a qualsiasi variabile numerica.

Una singola istruzione READ può accedere a una o più istruzioni DATA (in ordine), oppure diverse istruzioni READ possono accedere alla stessa istruzione DATA. Se il numero di variabili della lista supera il numero di elementi dell'istruzione (o delle istruzioni) DATA, viene visualizzato il messaggio d'errore "Out of data". Se il numero di variabili specificate è minore del numero di elementi dell'istruzione (o delle istruzioni) DATA, le istruzioni READ successive iniziano a leggere i dati a partire dal primo elemento non ancora letto. Se non ci sono istruzioni READ successive, i dati eccedenti vengono ignorati.

Per rileggere le istruzioni DATA dall'inizio, si deve usare l'istruzione RESTORE.

Esempio 1

```
.
.
.
80 FOR I=1 TO 10
90 READ A(I)
100 NEXT I
110 DATA 3.08,5.19,3.12,3.98,4.24
120 DATA 5.08,5.55,4.00,3.16,3.37
.
.
.
```

Questo segmento di programma legge i valori delle istruzioni DATA e li trasferisce nel vettore A. Dopo l'esecuzione il valore di A(1) è 3.08, il valore di A(2) è 5.19 e così via.

Esempio 2

```
LIST
10 PRINT "CITY","STATE","ZIP"
20 READ C$,S$,Z
30 DATA "DENVER,",COLORADO, 80211
40 PRINT C$,S$,Z
Ok
RUN
CITY        STATE        ZIP
DENVER,     COLORADO     80211
Ok
```

Questo programma legge dati stringa e numerici dall'istruzione DATA alla linea 30.

REM Istruzione

Permette l'inserimento in un programma di linee di commento.

REM *remark*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
remark	Rappresenta una sequenza di caratteri.

Caratteristiche

Le istruzioni REM non vengono eseguite, ma quando il programma viene listato esse vengono presentate esattamente come sono state impostate.

Un'istruzione GOTO e GOSUB può fare riferimento a un'istruzione REM; l'esecuzione continua dalla prima istruzione eseguibile dopo REM.

È possibile anche aggiungere campi commento alla fine della linea preceduti da un'apice ('); l'apice può anche essere digitato subito dopo il numero di linea.

Non si devono usare campi commento in un'istruzione DATA, perché essi verrebbero considerati come valori da assegnare.

Esempi

```
.
.
.
120 REM Velocità media calcolata
130 FOR I=1 TO 20
140 SUM=SUM +V(I)
150 NEXT I
.
.
.
```

oppure

```
.
.
.
120 FOR I=1 TO 20     'Velocità media calcolata
130 SUM=SUM+V(1)
140 NEXT I
.
.
.
```

oppure

```
.
.
.
120 'Velocità media calcolata
130 FOR I=1 TO 20
140 SUM=SUM+V(I)
150 NEXT I
.
.
.
```

RENUM Comando

Cambia i numeri di linea del programma corrente.

Il comando RENUM viene di solito usato in modo immediato.

RENUM [*newlinenum*][,[*oldlinenum*][,*increment*]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
newlinenum	È il primo numero di linea della nuova numerazione. Il valore di default è 10.
oldlinenum	È la linea del programma corrente da cui deve iniziare la numerazione. Il valore di default è la prima linea del programma.
increment	È l'incremento da usare nella nuova numerazione. Il valore di default è 10.

Caratteristiche

Il comando RENUM cambia anche tutti i numeri di linea che seguono le istruzioni GOTO, GOSUB, THEN, ON...GOTO, ON...GOSUB,

RESTORE, RESUME e ERL in modo da renderli congruenti con la nuova numerazione. Se in una di queste istruzioni compare un numero di linea non esistente, viene visualizzato il messaggio di errore "Undefined line nuber *xxxxx* in *yyyyy*". RENUM non corregge il numero di linea errato *xxxxx*, ma può cambiare il numero di linea *yyyyy*.

Il comando RENUM non può essere usato per cambiare l'ordine delle linee di programma o per creare numeri di linea maggiori di 65529. Si verifica in questo caso l'errore "Illegal function call".

Esempi

RENUM

Rinumera l'intero programma. Il primo nuovo numero di linea è 10 e viene assunto un incremento di 10.

RENUM 300,,50

Rinumera l'intero programma. Il primo nuovo numero di linea è 300 e viene assunto un incremento di 50.

RENUM 1000,900,20

Rinumera le linee da 900 in poi, iniziando così dal numero di linea 1000 e assumendo un incremento di 20.

RESET Comando

Chiude tutti i file dati aperti su tutti i drive.

Il comando RESET viene di solito usato all'interno di un programma.

RESET

Caratteristiche

Il comando RESET chiude tutti i file dati aperti su tutti i drive e provoca la registrazione su disco di tutti i blocchi in memoria. Perciò se manca la corrente, tutti i file vengono correttamente aggiornati. È necessario chiudere tutti i file prima di rimuovere un disco dal suo drive.

Se tutti i file dati aperti sono su disco, RESET agisce come l'istruzione CLOSE senza argomenti.

RESTORE Istruzione

Permette la rilettura di istruzioni DATA o dall'inizio del file dati interno o da una linea specificata.

L'istruzione RESTORE viene usata di solito all'interno di un programma.

RESTORE [*linenum*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
linenum	Deve essere il numero di linea di un'istruzione DATA.

Caratteristiche

Dopo l'esecuzione di un'istruzione RESTORE, l'istruzione READ successiva parte dal primo elemento della prima istruzione DATA del programma. Se *linenum* viene specificato, l'istruzione READ successiva legge a partire dal primo elemento dell'istruzione DATA specificata.

Esempio

```
10 READ A,B,C
20 RESTORE
30 READ D,E,F
40 DATA 58,67,97
 .
 .
 .
```

RESUME Istruzione

Riprende l'esecuzione di un programma dopo aver eseguito una routine di gestione errori.

L'istruzione RESUME viene di solito usata all'interno di un programma.

RESUME [**0** | **NEXT** | *linenum*]

Caratteristiche

SINTASSI	SIGNIFICATO
RESUME oppure RESUME 0	L'esecuzione riprende dall'istruzione che ha causato l'errore.
RESUME NEXT	L'esecuzione riprende alla prima istruzione successiva a quella che ha causato l'errore.
RESUME *linenum*	L'esecuzione riprende alla linea specificata.

Osservazioni

Può essere usato uno qualsiasi dei quattro formati illustrati sopra, a seconda del punto da cui l'esecuzione deve riprendere.

Un'istruzione RESUME non compresa in una routine di gestione errori, causa la visualizzazione del messaggio "RESUME without error".

Esempio

```
10 ON ERROR GOTO 900
 .
 .
 .
900 IF (ERR=230)AND(ERL=90) THEN PRINT "TRY
    AGAIN":RESUME 80
 .
 .
 .
```

RIGHT$ Funzione

Fornisce una sottostringa estraendo i caratteri più a destra da una stringa data.

RIGHT$(*string*,*length***)**

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
string	È un'espressione stringa il cui valore corrisponde alla stringa originale da cui la sottostringa deve essere estratta.
length	È un'espressione numerica, arrotondata all'intero più vicino, il cui valore, compreso tra 0 e 255, rappresenta la lunghezza della stringa richiesta.

Osservazioni

Se il parametro *length* è maggiore o uguale a LEN (*stringexp*) viene restituita l'intera stringa originale. Se *length* = 0 viene restituita la stringa nulla (di lunghezza zero).

Esempio

```
10 A$ = "DISK GWBASIC"
20 PRINT RIGHT$(A$,7)
RUN
GWBASIC
Ok
```

RMDIR Comando

Cancella una directory esistente.

RMDIR è generalmente usato in modo immediato.

RMDIR *pathname*

Dove

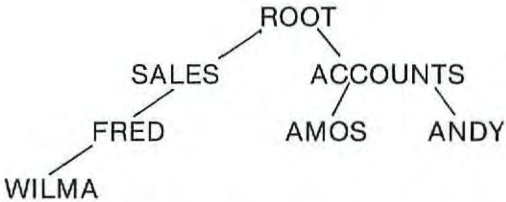
ELEMENTO DI SINTASSI	SIGNFICATO
pathname	È il nome della directory che deve essere cancellata.

Caratteristiche

RMDIR funziona esattamente come il comando RMDIR di MS-DOS.

La directory che deve essere cancellata deve essere svuotata di tutti i file e delle sottodirectory tranne la directory corrente (".") e gli ingressi alla directory madre ("..") altrimenti si verifica l'errore "Path not found".

Esempio



Con riferimento alla struttura vista sopra di suppone si voler eliminare la sottodirectory ANDY.

Supponendo che la directory corrente sia ROOT, allora:

RMDIR"ACCOUNTS\ANDY"

cancella la directory ANDY.

Se l'utente vuole far diventare ACCOUNTS la directory corrente e togliere la directory chiamata AMOS allora:

CHDIR "ACCOUNTS"
RMDIR "AMOS"

Errori Possibili

"Path not found"

"Bad file name": name della directory errato.

"Path/File Access Error": di solito indica che la directory non è vuota.

RND Funzione

Genera un numero casuale compreso tra 0 e 1.

RND [(*numexp*)]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
numexp	È un'espressione numerica che influenza il valore restituito.

Caratteristiche

La funzione RND restituisce un numero casuale uniformemente distribuito compreso tra 0 e 1 (estremi inclusi). La stessa sequenza di numeri casuali viene generata ogni volta che il programma viene eseguito, se non viene scritta un'istruzione RANDOMIZE prima della funzione RND.

La funzione RND svolge azioni diverse, a seconda del tipo di numero (positivo, negativo o uguale a zero) valutato dall'espressione numerica:

- RND (numero positivo) restituisce il numero successivo nella sequenza corrente.

- RND (numero negativo) riposiziona il generatore di numeri casuali e restituisce il primo numero casuale della nuova sequenza.

- RND (0) restituisce l'ultimo numero casuale generato, senza influenzare la sequenza corrente.

L'espressione numerica è opzionale, se non viene fornita RND agisce come se, come argomento, fosse stata fornita un'espressione positiva.

Per ottenere numeri interi casuali, compresi nell'intervallo da 0 a N, si deve usare INT (RND*(N + 1))

Esempio

```
10 FOR I=1 TO 5
20 PRINT INT(RND*100);
30 NEXT
RUN
 12  65  86  72  79
Ok
```

RUN Comando

Esegue il programma corrente o carica un programma da disco in memoria e lo esegue.

Sintassi 1

RUN [*linenum*]

Sintassi 2

RUN {*filespec* I *pathname*} [,R]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
linenum	È il numero di linea del programma residente in memoria. Se *linenum* viene specificato, l'esecuzione inizia da quella linea.
filespec (o *pahtname*)	È un'espressione di tipo stringa che specifica il programma da caricare in memoria ed eseguire.
R	Se viene specificata questa opzione, tutti i file dati (aperti prima di caricare in memoria il programma specificato) rimangono aperti.

Caratteristiche

Per un programma residente in memoria, se viene specificato il parametro *linenum* l'esecuzione del programma inizia da quella linea; altrimenti, l'esecuzione inizia dal numero di linea più basso.

Per eseguire un programma non residente in memoria è necessario specificare l'opzione *filespec* o *pathname*, con lo stesso nome di file usato per la registrazione.

Se il comando SAVE non fornisce l'estensione del nome del file, MS-DOS fornisce l'estensione di default .BAS. Prima di caricare in memoria il programma specificato, RUN chiude tutti i file dati aperti e cancella il contenuto corrente della memoria. Comunque, tramite l'opzione R, tutti i file dati rimangono aperti.

Esempi

RUN "B:NEWFILE",R

RUN A$

RUN 150

RUN "C:\R001\R002"

SAVE Comando

Registra il programma residente in memoria su disco e gli assegna un nome. Tramite l'opzione A il programma viene registrato in formato ASCII. Tramite l'opzione P esso viene registrato con protezione.

SAVE *filespec* | *pathname* **[,** {**A** | **P**}]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filespec (o *pathname*)	È un'espressione stringa che specifica il nome del file da salvare e, opzionalmente, il drive. Se viene omessa l'estensione

ELEMENTO DI SINTASSI	SIGNIFICATO
	del nome del file, viene assunto .BAS. Se viene omesso il drive, viene assunto il drive di default di MS-DOS.
A	Tale opzione registra il file programma in formato ASCII; altrimenti GW-BASIC registra il file in formato binario compatto.
P	Tale opzione registra il programma in formato binario codificato. Se un file protetto viene eseguito o caricato in memoria, ogni tentativo di listarlo o modificarlo fallisce.

Caratteristiche

Se sul disco selezionato esiste un file con lo stesso nome, questo viene cancellato e riscritto.

GW-BASIC registra il file in formato binario compresso ("tokenized"), a meno che non venga specificata l'opzione A.

Il formato ASCII occupa più spazio su disco, ma alcuni comandi richiedono che i file siano in formato ASCII, come ad esempio il comando MERGE e alcuni comandi di MS-DOS, come TYPE.

Se viene fatto il tentativo di usare il comando MERGE su programmi in binario, si verifica l'errore "Bad file mode".

Esempi

SAVE "SUPERB"

Registra il programma in memoria sul drive di default come SUPERB.BAS.

SAVE "A:PROG",A

Registra il programma in formato ASCII in memoria sul disco inserito nel drive A, come PROG.BAS.

SAVE "B:SECRET",P

Registra il programma in memoria in modo protetto sul disco inserito nel drive B, come SECRET.BAS.

SCREEN Funzione

Fornisce il codice ASCII (0-255) o il numero di colore del carattere situato alla locazione specificata su video.

SCREEN(*row*,*column*[,*condition*]**)**

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
row	È un'espressione numerica che restituisce un intero senza segno compreso tra 1 e 25.

ELEMENTO DI SINTASSI	SIGNIFICATO
column	È un'espressione numerica che restituisce un intero senza segno. L'intervallo di valori dipende dall'ampiezza del video.
condition	È un'espressione numerica, di confronto o logica che restituisce un valore booleano (0 oppure 1). Se viene specificato il parametro *condition* e non è uguale a zero, viene restituito il numero di colore del carattere invece del codice ASCII.

Caratteristiche

La funzione SCREEN, se usata in uno dei modi di grafica, restituisce zero se la locazione specificata contiene informazioni grafiche.

Per un elenco completo dei codici ASCII occorre far riferimento all'Appendice C.

Esempi

```
100 X = SCREEN (10,10)
```

Restituisce 65, se il carattere che si trova al punto (10,10) è A.

```
110 X = SCREEN (1,1,1)
```

Restituisce il numero di colore del carattere che si trova nell'angolo in alto a sinistra del video.

Errori Possibili

Nel caso vengano impostati valori che si trovano fuori dagli intervalli specificati si verifica l'errore ''Illegal function call''.

SCREEN Istruzione

Permette il passaggio da Modo Testo in uno dei modi di grafica. È possibile inoltre abilitare o disabilitare il colore e selezionare la pagina attiva e la pagina da visualizzare (in Modo Testo).

SCREEN [*mode*][,[*burst*][,[*apage*][,*vpage*]]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
mode	È un'espressione numerica il cui valore è un intero compreso tra 0 e 255. Essa definisce il Modo Testo (0), il Modo Grafica a Media Risoluzione (1), il Modo Grafica ad Alta Risoluzione (2) e il Modo Grafica ad Altissima Risoluzione (da 3 a 255).
burst	È un'espressione numerica, il cui valore è un intero uguale a 0 oppure a 1, che abilita il colore su un monitor non standard. Nel Modo Testo un valore uguale a 0 disabilita il colore e un valore uguale a 1 abilita il colore. In Media

ELEMENTO DI SINTASSI	SIGNIFICATO
	Risoluzione un valore uguale a 0 abilita il colore e un valore uguale a 1 lo disabilita. In Alta e ad Altissima Risoluzione il parametro *burst* viene ignorato, poiché questi due modi funzionano solo in B/N. Questo parametro non ha alcun significato se si usa un monitor standard.
apage (solo Modo Testo)	È un'espressione intera compresa nell'intervallo da 0 a 7, per un'ampiezza video di 40, e da 0 a 3, per un'ampiezza video di 80. Essa seleziona la "pagina attiva", cioé la pagina su cui si scrive tramite istruzioni di output su video. Se *apage* viene omesso, la pagina attiva assume il valore di default 0.
vpage (solo Modo Testo)	È un'espressione intera compresa nell'intervallo da 0 a 7, per un'ampiezza video di 40, e da 0 a 3, per un'ampiezza video di 80. Essa seleziona la "pagina visiva", cioé la pagina da visualizzare, che può essere diversa dalla pagina attiva. Se questo parametro viene omesso, la pagina visiva assume come valore di default la pagina attiva.

Parametri *mode* e *burst*

Nella tabella che segue le prime due colonne rappresentano i parametri *mode* e *burst* di un'istruzione SCREEN.

Il parametro *burst* abilita il colore su monitor non standard a colori. Questo parametro non ha un effettivo significato in sistemi con monitor standard; cioé l'assegnazione del valore 0 o 1 a *burst* in Media Risoluzione, ha lo stesso effetto se viene usato un monitor standard a colori o un monitor standard in bianco e nero (in questo caso i quattro colori appaiono come toni di grigio).

mode	*burst*	**DESCRIZIONE**
0	0	80 c. × 25 r. - Modo Testo B/N.
0	1	80 c. × 25 r. - Modo Testo Colore.
1	0	320 pixel orizzontali × 200 pixel verticali. Modo Grafica a Media Risoluzione Colore (40 c. × 25 r.).
1	1	320 pixel orizzontali × 200 pixel verticali. Modo Grafica a Media Risoluzione B/N (40 c. × 25 r.).
2	X (ignorato)	640 pixel orizzontali × 200 pixel verticali. Modo Grafica ad Alta Risoluzione B/N (80 c. × 25 r.).
3-255	X (ignorato)	640 pixel orizzontali × 400 pixel verticali. Modo Grafica ad Altissima Risoluzione B/N (80 c. × 25 r.).

Tab. 8-3 Rappresentazione dei Parametri mode e burst

Valori di Default

Se non viene impostata un'istruzione SCREEN, il sistema assume i seguenti valori di default:

mode	=	0 (Modo Testo)
burst	=	0 (B/N)
apage	=	0 (pagina attiva 0)
vpage	=	0 (pagina visiva 0)

L'istruzione SCREEN deve precedere tutte le istruzioni di I/O sul video, ma è possibile usare più di un'istruzione SCREEN per definire diversi attributi video per diverse sezioni del programma.

Parametri *apage* e *vpage*

Se viene selezionato il Modo Testo, è possibile specificare due ulteriori parametri (*apage* e *vpage*) per selezionare la pagina attiva e visiva. Ci sono otto pagine di visualizzazione (numerate da 0 a 7) nel Modo Testo a 40 colonne e quattro pagine di visualizzazione (numerate da 0 a 3) nel Modo Testo a 80 colonne. In ciascuno dei tre modi grafici è disponibile una sola pagina di visualizzazione.

Vi è un solo cursore per tutte le pagine di visualizzazione, perciò, se viene selezionata una nuova pagina attiva, è necessario salvare la posizione del cursore (tramite POS(0) e CSRLIN) prima di passare alla nuova pagina. Quando si ritorna alla pagina precedente bisogna riposizionare il cursore tramite l'istruzione LOCATE (Testo).

Se l'istruzione SCREEN è usata al solo scopo di cambiare le pagine, i primi due parametri (*mode* e *burst*) possono essere omessi.

Ampiezza del Video

Dopo l'inizializzazione l'ampiezza video è di 80 colonne, pertanto può venir usata l'istruzione WIDTH per selezionare il video di 40 colonne. Se, tramite l'istruzione SCREEN, si seleziona il modo a Media Risoluzione, il numero di colonne è fissato a 40 (senza usare l'istruzione WIDTH).

Nel Modo Testo, l'istruzione WIDTH può essere usata per selezionare il modo a 40 colonne o quello a 80 colonne. L'istruzione WIDTH può venir usata anche per selezionare i modi 1 o 2 (modo a Media o Alta Risoluzione).

Selezionando il Modo Testo (*mode* =0), dopo aver selezionato uno dei modi di grafica, verrà definito un video a 40 colonne o un video a 80 colonne, a seconda dell'ampiezza usata nel modo grafico.

Per esempio:

SCREEN 1 Definisce il Modo a Media Risoluzione (video con 40 colonne).

SCREEN 0 Definisce il Modo Testo (l'ampiezza del video è sempre di 40 colonne).

Osservazioni

SE...	ALLORA...
tutti i parametri sono validi	viene memorizzato il nuovo modo video, il video è cancellato, sono ripristinati i valori di default per i colori di fondo e di primo piano.
tutti i parametri sono identici a quelli precedenti	niente è cambiato.
un parametro viene omesso	viene assunto il valore precedente (fatta eccezione per la pagina visiva che viene posta uguale alla pagina attiva).

Esempi

```
10 SCREEN 0,1,0,0
```

Seleziona il Modo Testo a colori, sia la pagina attiva che quella visiva sono poste uguale a 0.

```
20 SCREEN ,,1,2
```

I parametri *mode* e *burst* rimangono invariati, la pagina attiva è posta a 1, mentre quella visiva è posta a 2.

```
30 SCREEN 2
```

Viene selezionato il Modo Grafica ad Alta Risoluzione.

```
40 SCREEN 1,1
```

Viene selezionato il Modo Grafica a Media Risoluzione a colori.

```
50 SCREEN ,0
```

Viene disabilitato il colore, rimanendo in Modo Grafica a Media Risoluzione.

Errori Possibili

Se viene usato un valore fuori dagli intervalli specificati, viene emesso il messaggio "Illegal function call".

SGN Funzione

Restituisce un valore in base al segno dell'espressione numerica specificata.

SGN(*numexp*)

Caratteristiche

Se *numexp*>0, SGN(*numexp*) restituisce 1.
Se *numexp*=0, SGN(*numexp*) restituisce 0.
Se *numexp*<0, SGN(*numexp*) restituisce -1.

Esempio

```
50 ON SGN(X)+2 GOTO 300,400,500
```

salta a 300 se *numexp* è negativo, a 400 se *numexp* è uguale a zero e a 500 se *numexp* è positivo.

SHELL Comando

Carica in memoria ed esegue un altro programma (.EXE o .COM o .BAT).

SHELL [*stringexp*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
stringexp	È un'espressione stringa che contiene il nome di un programma da eseguire e (opzionalmente) gli argomenti del comando.

Caratteristiche

Quando il programma termina, il controllo ritorna al programma GW-BASIC, all'istruzione che segue il comando SHELL. Un programma eseguito sotto il controllo di GW-BASIC viene denominato "un processo child".

I processi child vengono eseguiti caricando ed eseguendo tramite SHELL una copia di COMMAND con lo switch /C. Usando COMMAND in questo modo, i parametri della linea di comando vengono passati al processo child. È possibile selezionare l'input e l'output standard ed eseguire i comandi di sistema DIR, PATH e SORT.

Osservazioni

Il nome del programma contenuto in *stringexp* può avere un'estensione qualsiasi, poiché COMMAND ne effettua il controllo. Se non viene fornita nessuna estensione, COMMAND cerca un file .COM, poi un file .EXE ed infine un file .BAT. Se COMMAND non viene trovato, SHELL emette l'errore "File not found" ;non viene generato nessun errore se COMMAND non trova il file specificato in *stringexp*.

Qualsiasi testo in *stringexp*, separato del nome del programma da almeno 1 spazio, viene trattato da COMMAND come sequenza di parametri del programma.

GW-BASIC rimane in memoria mentre il processo child è in corso di esecuzione; quando tale esecuzione termina, il controllo ritorna a GW-BASIC.

SHELL senza *stringexp* fornisce all'utente un nuovo COMMAND, consentendogli di eseguire una qualunque delle sue funzioni. Per tornare a GW-BASIC si deve usare il comando EXIT di MS-DOS.

Esempi

SHELL	(fornisce un nuovo COMMAND)
A>**DIR**	(si imposta DIR per vedere i file)
A>**EXIT**	(si imposta EXIT per ritornare a GW-BASIC)

L'esempio seguente scrive alcuni dati da ordinare, SHELL esegue SORT per ordinarli, poi legge i dati ordinati per scrivere un report.

```
900 OPEN "SORTIN.DAT" FOR OUTPUT AS #1
 ...
950 REM Scrive i dati da ordinare
 ...
1000 CLOSE 1
1010 SHELL "SORT <SORTIN.DAT>SORTOUT.DAT"
1020 OPEN "SORTOUT. DAT" FOR INPUT AS #1
1030 REM Elabora i dati ordinati.
 ...
10 SHELL "DIR I SORT>FILES.
20 OPEN "FILES." FOR INPUT AS #1
 ...
```

Errori Possibili

"File not found": non è stato trovato il nome del file (.EXE o .COM o.BAT).

"Out of memory": la memoria disponibile non è sufficiente per eseguire il processo child.

"Can't continue after SHELL": la memoria disponibile non è sufficiente affinché GW-BASIC possa continuare. Tutti i file sono chiusi e GW-BASIC ritorna a MS-DOS. Questo può succedere quando l'esecuzione di un processo child che rimane residente termina.

"Internal error": GW-BASIC o MS-DOS non funzionano correttamente.

SIN Funzione

Calcola il seno dell'argomento.

SIN(*numexp*)

Caratteristiche

La funzione SIN viene calcolata in semplice precisione se non viene specificato lo switch /D impostando il comando GWBASIC.

Esempio

```
PRINT SIN(1.5)
 .9974951
Ok
```

SOUND Istruzione

Produce un suono.

SOUND *frequency,duration*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
frequency	È un'espressione numerica compresa tra 37 e 32767 che rappresenta la frequenza in Hertz.
duration	È la durata espressa in periodi di clock che si verificano 18.2 volte al secondo; *duration* è un'espressione intera compresa tra da 0 e 65535.

Caratteristiche

Se la durata è ugale a zero, qualsiasi istruzione SOUND in corso di esecuzione viene disabilitata. Un'istruzione SOUND con durata uguale a zero non ha effetto se non vi è nessuna istruzione SOUND in corso di esecuzione.

Di seguito sono visualizzate le frequenze delle note musicali (due ottave sotto e due ottave sopra il C centrale):

1975.5	1760.0	1568.0	1396.9	1318.5	1174.7	1046.5
B	A	G	F	E	D	C
987.77	880.00	783.99	698.46	659.26	587.33	523.25
B	A	G	F	E	D	C
493.88	440.00	392.00	349.23	329.63	293.66	261.63
B	A	G	F	E	D	C
246.94	220.00	196.00	174.61	164.81	146.83	130.81
B	A	G	F	E	D	C

TEMPI	BATTUTE/ MINUTO	PERIODI DI CLOCK/ BATTUTE
Larghissimo Largo Larghetto Grave Lento Adagio	 40-60 60-66 66-76	 28.13-18.75 18.75-17.05 17.05-14.8
Adagietto Andante	 76-108	 14.8-10.42
Antantino Moderato	 108-120	 10.42-9.38

Tab. 8-4 Tempi di Battute/Minuto (cont.)

TEMPI	BATTUTE/ MINUTO	PERIODI DI CLOCK/ BATTUTE
Allegretto Allegro Vivace Veloce Presto	120-168 168-208	9.38-6.7 6.7-5.41
Prestissimo	maggiore di 208	minore di 5,41

Tab. 8-4 Tempi di Battute/Minuto

Esempio

100 SOUND RND*1000+37,2

Questa istruzione crea suoni casuali.

SPACE$ Funzione

Fornisce una stringa di spazi della lunghezza richiesta.

SPACE$(*length*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
length	È un'espressione intera compresa tra 0 e 255 e specifica il numero di spazi, cioé la lunghezza della stringa che deve essere restituita.

Esempio

```
10 FOR I=1 TO 5
20 X$=SPACE$(I)
30 PRINT X$;I
40 NEXT I
RUN
 1
  2
   3
    4
     5
Ok
```

Errori Possibili

Se il parametro *length* si trova fuori dall'intervallo specificato, si verifica l'errore "Illegal function call".

SPC Funzione

Inserisce *n* spazi in un'istruzione PRINT, LPRINT o PRINT#.

SPC(*n*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È un'espressione intera, compresa tra 0 e 255, che specifica il numero di spazi da inserire nella linea visualizzata.

Osservazioni

La funzione SPC può essere usata con le istruzioni PRINT, LPRINT e PRINT#.

Se *n* è maggiore dell'ampiezza definita, il valore usato è *n* MOD *width*. La funzione SPC deve essere seguita da un punto e virgola (;), perciò GW-BASIC non aggiunge un ritorno a capo se la funzione è alla fine della lista di dati.

Se *n* si trova fuori dall'intervallo specificato, si verifica l'errore "Illegal function call".

Esempi

```
PRINT "FOUR" SPC(15) "SEASONS"
FOUR               SEASONS
Ok
```

SQR Funzione

Calcola la radice quadrata di un'espressione numerica positiva.

SQR(*numexp*)

Osservazioni

La funzione SQR viene calcolata in semplice precisione, se non viene specificata l'opzione /D impostando il comando GWBASIC.

Se l'argomento è negativo si verifica l'errore "Illegal function call".

Esempio

```
10 FOR X=10 TO 25 STEP 5
20 PRINT X, SQR(X)
30 NEXT
RUN
 10        3.162278
 15        3.872984
 20        4.472136
 25        5
Ok
```

STOP Istruzione

Interrompe l'esecuzione del programma e ritorna in stato comandi.

L'istruzione STOP viene usata solo all'interno di un programma.

STOP

Caratteristiche

L'istruzione STOP può essere usata in qualsiasi punto del programma. In questo caso viene visualizzato il seguente messaggio:

Break in *nnnnn*

A differenza dell'istruzione END, l'istruzione STOP non chiude i file.

GW-BASIC ritorna sempre in stato comandi dopo l'esecuzione dell'istruzione STOP; è possibile riprendere l'esecuzione tramite il comando CONT.

Esempio

```
10 INPUT A,B,C
20 K=A^2*5.3:L=B^3/.26
30 STOP
40 M=C*K+100:PRINT M
RUN
? 1,2,3
Break in 30
```

```
Ok
PRINT L
 30.76923
Ok
CONT
 115.9
Ok
```

STR$ Funzione

Converte in formato stringa il valore di un'espressione numerica.

STR$(*numexp*)

Esempio 1

```
10 A$ = STR$(70)
20 PRINT A$
Ok
RUN
70
Ok
```

70 (argomento di STR$) è un numero, ma il contenuto della variabile A$ è una stringa di due caratteri il cui valore è 70.

Esempio 2

```
 5 REM MATEMATICA PER I RAGAZZI
10 INPUT "TYPE A NUMBER"; N
20 ON LEN(STR$(N)) GOSUB 30,100,200,300,400,500
  .
  .
  .
```

Il numero N, introdotto da tastiera, è convertito in stringa tramite la funzione STR$.

Esempio 3

```
10 A = 1.3
20 A = VAL(STR$(A))
30 PRINT A
Ok
RUN
 1.3
Ok
```

VAL è la funzione complementare di STR$.

STRING$ Funzione

Crea una stringa di lunghezza specificata, i cui caratteri hanno tutti lo stesso codice ASCII o sono uguali al primo carattere di una data stringa.

Sintassi 1

STRING$(*length,code***)**

Sintassi 2

STRING$(*length,stringexp***)**

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
length	È un'espressione intera compresa tra 0 e 255, che specifica la lunghezza della stringa risultante.
code	È un'espressione intera compresa tra 0 e 255, che specifica il codice ASCII, il cui carattere corrispondente viene utilizzato per formare la stringa risultante.
stringexp	È un'espressione stringa, il cui primo carattere viene utilizzato per formare la stringa risultante.

Esempio 1

```
10 X$ = STRING$(10,45)
20 PRINT X$; "MONTHLY REPORT"; X$
RUN
---------- MONTHLY REPORT ----------
Ok
```

Esempio 2

```
10 LET A$ = "HOUSTON"
20 LET X$ = STRING$(8,A$)
30 PRINT X$
RUN
HHHHHHHH
```

SWAP Istruzione

Permette di scambiare i valori di due variabili.

SWAP *variable1,variable2*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
variable1 e *variable2*	Sono due variabili dello stesso tipo (intero, in semplice precisione, in doppia precisione o stringa).

Osservazioni

Le due variabili i cui valori vengono scambianti devono essere dello stesso tipo, altrimenti viene visualizzato l'errore "Type mismatch". Se, quando viene eseguita l'istruzione SWAP, la seconda variabile non è ancora stata definita si verifica l'errore "Illegal function call".

Esempio

```
Ok
10 A$ = " ONE ": B$ = " ALL ": C$ = "FOR"
20 PRINT A$ C$ B$
30 SWAP A$, B$
40 PRINT A$ C$ B$
```

```
RUN
ONE FOR ALL
ALL FOR ONE
Ok
```

Dopo l'esecuzione della linea 30, A$ assume il valore " ALL " e B$ assume il valore " ONE ".

SYSTEM Comando

Chiude tutti i file dati aperti e restituisce il controllo a MS-DOS.

SYSTEM

Caratteristiche

Quando viene eseguito il comando SYSTEM tutti i file dati aperti vengono chiusi, il programma GW-BASIC viene perso e il controllo viene restituito a MS-DOS. Se si è entrati in ambiente GW-BASIC da MS-DOS, tramite un file Batch, il comando SYSTEM restituisce il controllo al file Batch.

TAB Funzione

In un'istruzione PRINT, LPRINT o PRINT # la funzione TAB posiziona il cursore o la testina della stampante alla posizione specificata.

TAB(*n*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È un'espressione intera compresa tra 1 e 255.

Caratteristiche

Se la posizione del cursore o della testina della stampante è oltre la posizione specificata dal valore *n*, la funzione TAB posiziona il cursore o la testina nella giusta posizione sulla linea successiva. 1 rappresenta il limite sinistro mentre il limite destro è dato dall'ampiezza definita meno uno.

Se il valore di *n* supera l'ampiezza definita, viene applicata l'operazione modulo. Per esempio PRINT TAB(243) su un video di 40 colonne, ha lo stesso effetto di PRINT TAB(3), perché 243 MOD 40 = 3.

La funzione TAB è sempre seguita da un punto e virgola, perciò GW-BASIC non aggiunge un ritorno a capo se la funzione TAB si trova alla fine della lista dei dati.

Esempio

```
10 PRINT "NAME" TAB(25) "AMOUNT":PRINT
20 READ A$, B$
30 PRINT A$ TAB(25) B$
40 DATA "G. T. JONES","$25.00"
RUN
NAME                            AMOUNT

G. T. JONES                     $25.00
Ok
```

TAN Funzione

Calcola la tangente dell'argomento.

TAN(*numexp*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
numexp	È un'espressione numerica che rappresenta l'angolo in radianti.

Caratteristiche

La funzione TAN (*numexp*) viene calcolata in semplice precisione se non viene specificata l'opzione /D impostando il comando GWBASIC.

Se nell'esecuzione di TAN si verifica un overflow, viene visualizzato il messaggio di errore "Overflow", viene fornito come risultato il numero massimo che la macchina può rappresentare, con il segno appropriato, e l'esecuzione continua.

Esempio

```
PRINT TAN(3.14/4)
 .999204
Ok
```

TIME$ Istruzione e Funzione

L'istruzione TIME$ permette di impostare l'ora.

La funzione TIME$ permette di leggere l'ora corrente.

Sintassi 1: istruzione

TIME$ = *stringexp*

Sintassi 2: funzione

stringvar = **TIME$**

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
stringexp	È un'espressione stringa che indica l'ora da impostare.
stringvar	È una variabile stringa che riporta l'ora corrente (stringa di 8 caratteri).

Caratteristiche

Usata come istruzione, *stringexp* è un'espressione stringa indicante l'ora in una delle seguenti forme:

- *hh* (imposta l'ora; i minuti e i secondi assumono il valore di default di 00)
- *hh:mm* (imposta l'ora e i minuti; i secondi assumono il valore di default di 00)
- *hh:mm:ss* (imposta l'ora, i minuti e i secondi)

Viene usato un intervallo di 24 ore; cioé per esempio, l'ora 8:00 p.m. deve essere impostata come 20:00:00.

È possibile omettere lo zero prima dei valori appena descritti, ma è necessario in questo caso impostare almeno una cifra. L'ora può anche essere stata impostata da MS-DOS prima di entrare in ambiente GW-BASIC.

Usata come funzione, TIME$ restituisce una stringa di otto caratteri espressa nella forma *hh:mm:ss*, dove *hh* rappresenta l'ora (da 00 a 23), *mm* rappresenta i minuti (da 00 a 59) e *ss* rappresenta i secondi (da 00 a 59).

Esempio 1

```
TIME$ = "8:0"
Ok
PRINT TIME$
08:00:04
Ok
```

Esempio 2

Il programma che segue permette di visualizzare la data e l'ora corrente sulla 25 linea del video e di far funzionare il segnale orario.

```
10 KEY OFF:SCREEN 0:WIDTH 40:CLS
20 LOCATE 25,5
30 PRINT DATE$,,TIME$
40 SEC=VAL(MID$(TIME$,7,2))
50 IF SEC = SSEC THEN 20 ELSE SSEC = SEC
60 IF SEC = 0 THEN 1010
70 IF SEC = 30 THEN 1020
80 IF SEC < 57 THEN 20

1000 SOUND 1000,2:GOTO 20
1010 SOUND 2000,8:GOTO 20
1020 SOUND 400,4:GOTO 20
```

Errori Possibili

Se i valori non sono compresi nell'intervallo specificato si verifica l'errore "Illegal function call". Viene mantenuta l'ora precedente.

Se *stringexp* non è una stringa valida si verifica l'errore "Type mismatch".

TIMER Funzione

Restituisce un numero in semplice precisione che indica quanti secondi sono trascorsi dalla mezzanotte o dal reset del sistema.

TIMER

Caratteristiche

TIMER è una funzione numerica che calcola i secondi frazionari al grado più vicino possibile. Non può essere usata come una variabile utente.

Esempio

```
10 FOR K = 1 TO 10
20 PRINT "TIMER ="; TIMER
30 NEXT
```

TIMER Istruzione

TIMER ON abilita il rilevamento di eventi del TIMER.
TIMER OFF disabilita il rilevamento di eventi del TIMER.
TIMER STOP sospende il rilevamento di eventi del TIMER.

TIMER {ON | OFF | STOP}

Caratteristiche

L'istruzione TIMER ON abilita un rilevamento di eventi in tempo reale tramite l'istruzione ON TIMER(*n*) GOSUB.

Durante il rilevamento abilitato dall'istruzione ON TIMER(*n*) GOSUB, GW-BASIC controlla tutte le istruzioni per vedere se il timer ha raggiunto il livello specificato. In caso affermativo l'istruzione ON TIMER(*n*) GOSUB viene eseguita.

TIMER OFF disabilita il rilevamento di eventi. Se si verifica un evento non viene rilevato da una eventuale istruzione TIMER ON.

TIMER STOP disabilita il rilevamento di eventi. Se si verifica un evento viene memorizzato e l'istruzione ON TIMER(*n*) GOSUB viene eseguita non appena viene abilitato il rilevamento.

Quando è eseguita l'istruzione ON TIMER(*n*) GOSUB viene eseguito un TIMER STOP automatico. Quando ha luogo un rilevamento di errore tutti i rilevamenti vengono disabilitati

TRON/TROFF Comandi

Il comando TRON (TRACE ON) permette di listare il numero di linea di ogni istruzione eseguita.

Il comando TROFF (TRACE OFF) interrompe il listato dei numeri di linea iniziato tramite il comando TRON.

Sintassi 1

TRON

Sintassi 2

TROFF

Caratteristiche

Il comando TRON (eseguito in modo immediato o all'interno di un programma) viene usato come strumento di debugging. Se TRON è attivo ogni linea i programma viene visualizzata quando viene eseguita. I numeri di linea sono racchiusi tra parentesi quadre. L'indicatore di traccia viene disabilitato tramite il comando TROFF (o quando viene eseguito un comando NEW).

Esempio

```
TRON
Ok
LIST
10 K=10
20 FOR J=1 TO 2
30 L=K+10
40 PRINT J;K;L
50 K=K+10
60 NEXT
70 END
Ok
RUN
[10][20][30][40] 1   10   20
[50][60][30][40] 2   20   30
[50][60][70]
Ok
TROFF
Ok
```

UNLOCK Istruzione

L'istruzione UNLOCK elimina la protezione ad un file aperto. L'istruzione può essere usata solo se è stato installato MS-NET con MS-DOS 3.1 o future versioni.

UNLOCK [#] *filenum* [,[*recordnum1*] [**TO** *recordnum2*]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filenum	È il numero del file aperto.
recordnum1	È il numero del record dal quale si parte a togliere la protezione. Se questo parametro è omesso, viene assunto il record numero 1 come inizio.
recordnum2	È il numero dell'ultimo record su cui è stata tolta la protezione. Se questo parametro è omesso, viene tolta la protezione solo sul record espresso da *recordnum1*.

Caratteristiche

Se è stato specificato un record o un intervallo di record, e il file è stato aperto in modo random, solo a questi record sarà tolta la protezione.

L'intervallo di record deve essere esattamente uguale a quello definitivo nell'istruzione LOCK, altrimenti si incorrerà in un errore di "Permission Denied".

Vedere le istruzioni OPEN e LOCK.

Nota

Se l'utente tenta di usare questa istruzione con release MS-DOS precedenti alla 3.1, incorrerà in un "Advanced Feature Error". Se si sta usando MS-DOS 3.1 o future versioni, ma MS-NET non è stato installato, si incorrerà in un errore di "Permission Denied". Si deve introdurre il comando SHARE di MS-DOS (o alla comparsa del prompt di MS-DOS, o in un file AUTOEXEC.BAT).

Vedere il manuale "MS-DOS Guida Utente" per una completa descrizione delle procedure da seguire per la rete.
Il modo consigliato per utilizzare file su device condivisi, è quello di eseguire un LOCK su un file o un intervallo di record all'interno di un file, prima di tentare di leggere o scrivere su quel file. È anche consigliabile eseguire un UNLOCK su quel file o intervallo di record, prima che questi venga chiuso. Se questo non viene fatto, si potrebbero verificare dei problemi nell'accesso a quel file in futuro, in ambiente di rete.

Il tempo di LOCK dovrebbe essere mantenuto al minimo.

Esempi

```
LOCK #1,1 TO 4
LOCK #1,5 TO 8
.
.
UNLOCK #1,1 TO 4
UNLOCK #1,5 TO 8
```

Notare che le istruzioni UNLOCK riportate qui sopra non possono essere rimpiazzate da un'unica istruzione:

```
UNLOCK #1,1 TO 8
```

Errori Possibili

"Permission Denied": MS-NET non installato o intervallo di record non esattamente uguale a quello definito nell'istruzione LOCK.

USR Funzione

Richiama un sottoprogramma scritto in linguaggio macchina.
La funzione USR viene di solito usata all'interno di un programma.

USR[*n*] (*argument*)

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
n	È un'intero compreso tra 0 e 9 e specifica il sottoprogramma USR che viene richiamato. Non devono essere inseriti spazi tra USR e *n*. Se *n* viene omesso, viene assunto USR0.
argument	È il valore passato al sottoprogramma e può essere qualsiasi espressione numerica o stringa. Anche se il sottoprogramma non richiede un argomento, deve essere fornito un argomento fittizio.

Caratteristiche

Il tipo della variabile (numerico o stringa) che riceve la chiamata di funzione deve essere conforme a quello dell'argomento passato.

Prima di richiamare qualsiasi funzione USR deve essere eseguita l'istruzione DEF USR corrispondente, per definire lo spiazzamento del sottoprogramma riferito all'indirizzo iniziale del segmento, specificato dall'ultima istruzione DEF SEG eseguita. Se non è stata eseguita nessuna istruzione DEF SEG viene usato l'indirizzo iniziale del Segmento Dati GW-BASIC.

L'istruzione CALL rappresenta un'altro modo di richiamare un sottoprogramma in linguaggio macchina.

Esempio

```
100 DEF SEG = &H8000
110 DEL USR0 = 0
120 X = 5
130 Y = USR0(X)
140 PRINT Y
```

VAL Funzione

Converte la rappresentazione stringa di un numero nel suo valore numerico.

VAL(*stringexp*)

Caratteristiche

La funzione VAL elimina gli spazi iniziali, i caratteri di tabulazione e di interlinea dall'espessione stringa.

La stringa risultante, se è una rappresentazione numerica valida, viene convertita in un numero, altrimenti VAL restituisce 0 (zero). Per esempio:

VAL ("-3")

restituisce -3.

VAL ("ABC")

restituisce 0.

Esempio

```
10 READ NAME$,CITY$,STATE$,ZIP$
20 IF VAL(ZIP$)<90000 OR VAL(ZIP$)>96699
   THEN PRINT NAME$ TAB(25) "OUT OF STATE"
30 IF VAL(ZIP$)> =90801 AND VAL(ZIP$)< =90815
   THEN PRINT NAME$ TAB(25) "LONG BEACH"
.
.
.
```

VARPTR Funzione

VARPTR (*variable*) fornisce l'indirizzo di memoria di *variable*.

VARPTR (# *filenum*) fornisce l'indirizzo iniziale del File Control Block (FCB) associato al file specificato da *filenum*.

Sintassi 1

VARPTR(*variable***)**

Sintassi 2

VARPTR(# *filenum***)**

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
variable	È il nome di una variabile numerica o stringa definita nel programma.
filenum	È il numero con cui il file è stato aperto.

Osservazioni

Per entrambe le sintassi, l'indirizzo restituito è un intero compreso tra -32768 e 32767. Questo numero esprime lo spiazzamento del Segmento Dati GW-BASIC.

Sintassi 1

Fornisce l'indirizzo di memoria del primo byte del dato identificato da *variable.*

Prima di eseguire VARPTR è necessario assegnare un valore a *variable*, altrimenti si verifica l'errore "Illegal function call". È possibile usare qualsiasi tipo di variabile (numerico o stringa); per le variabili stringa viene fornito l'indirizzo del primo byte del descrittore di stringa.

La funzione VARPTR viene di solito usata per ottenere l'indirizzo di una variabile o di una matrice, affinché possa essere passata ad un sottoprogramma in linguaggio macchina. Viene di solito

specificata una chiamata di funzione nella forma VARPTR (A(0)) per ottenere l'indirizzo del primo elemento della matrice A.

Prima di eseguire VARPTR per una matrice devono essere assegnate tutte le variabili semplici, perché gli indirizzi delle matrici cambiano ogni volta che viene assegnato un nuovo valore a una variabile semplice.

Sintassi 2

VARPTR (#*filenum*) restituisce l'indirizzo iniziale del File Control Block (FCB).

Esempi

```
10 X = USR (VARPTR(Y))

100 FIELDADDRESS = VARPTR(#2)
```

VARPTR$ Funzione

Restituisce, in formato stringa, l'indirizzo di memoria della variabile.

VARPTR$(*variable*)

Caratteristiche

VARPTR$ viene fondamentalmente usata per eseguire sottostringhe contenenti le istruzioni DRAW e PLAY, in programmi che verranno successivamente compilati. Con programmi che non verranno successivamente compilati, la sintassi standard delle

istruzioni DRAW e PLAY è sufficiente a produrre gli effetti desiderati.

Per esempio:

PLAY "XA$;"

e

PLAY "X" + VARPTR$(A$)

producono lo stesso effetto.

Prima di eseguire VARPTR$ è necessario assegnare un valore a *variable* altrimenti si verifica l'errore "Illegal function call". Le variabili vengono definite tramite un qualunque riferimento alla variabile. È possibile usare qualsiasi tipo di variabile (numerico o stringa).

VARPTR$ restituisce una stringa di tre byte nella forma:

byte 0 = tipo della variabile
byte 1 = byte meno significativo dell'indirizzo
byte 2 = byte più significativo dell'indirizzo

Il tipo della variabile può essere:

2 intero
3 stringa
4 semplice precisione
8 doppia precisione

Poiché gli indirizzi delle matrici cambiano ogni volta che viene definita una nuova variabile semplice, non è consigliabile salvare il risultato di una funzione VARPTR$ in una variabile. È raccomandabile eseguire VARPTR$ prima di usare il risultato.

VIEW Istruzione

Definisce le dimensioni e là posizione della finestra fisica entro cui visualizzare una figura (solo modo grafica).

VIEW [[**SCREEN**] [(*vx1* , *vy1*)-(*vx2* , *vy2*) [,[*color*] [,*border*]]]]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
(*vx1*, *vy1*) (*vx2*, *yx2*)	(*vx1*, *vy1*) sono le coordinate superiori di sinistra e (*vx2*, *vy2*) le coordinate inferiori di destra della finestra fisica.
color	Permette di colorare la finestra fisica con un colore specificato. Se *color* viene omesso, la finestra non viene colorata.
border	Permette di tracciare il contorno della finestra fisica (se è disponibile lo spazio necessario). Se *border* viene omesso il contorno non viene tracciato.

Caratteristiche

VIEW definisce un'area rettangolare su video, detta finestra fisica, entro cui visualizzare una figura, specificando le coordinate superiori di sinistra (*vx1*, *vy1*) e le coordinate inferiori di destra (*vx2*, *vy2*).

L'istruzione VIEW senza parametri definisce l'intero video come finestra.

Opzione SCREEN

L'opzione SCREEN definisce le coordinate *x* e *y* di tutti i punti come assolute rispetto al video (origine nel vertice superiore sinistro del video) ma solo i punti all'interno della finestra fisica sono illuminati.

Utilizzando il formato:

VIEW (*vx1*, *vy1*) - (*vx2*, *vy2*)

le coordinate *x* e *y* di tutti i punti sono relative alla finestra fisica. Cioé, se vengono impostate le istruzioni:

VIEW (10,10)-(200,100)
PSET (0,0),3

il punto riferito dall'istruzione PSET si trova effettivamente alla posizione (10,10) su video. Così pure se viene eseguita un'istruzione WINDOW, tutti i punti all'interno della finestra logica vengono proiettati sui corrispondenti punti all'interno della finestra fisica definita da VIEW.

Utilizzando il formato:

VIEW SCREEN (*vx1*, *vy1*)-(*vx2*, *vy2*)

le coordinate *x* e *y* di tutti i punti sono assolute. Cioé se vengono impostate le istruzioni:

VIEW SCREEN (10,10)-(200,100)
PSET (0,0),3

il punto riferito da PSET non compare perché il punto (0,0) si trova fuori dalla finestra fisica; con l'istruzione PSET (10,10),3 viene illuminato il punto nel vertice superiore sinistro della finestra. Se viene eseguita un'istruzione WINDOW, tutti i punti all'interno della finestra logica vengono proiettati sui corrispondenti punti del video, ma sono visibili solo quelli all'interno della finestra fisica definita da VIEW SCREEN.

Finestre Multiple

Può essere eseguito un qualsiasi numero di istruzioni VIEW. Ad ogni esecuzione dell'istruzione VIEW viene definita una finestra fisica che è la finestra corrente. Per cambiare tale finestra è necessario eseguire un'altra istruzione VIEW.

Le finestre fisiche possono anche essere totalmente o parzialmente sovrapposte. Per cancellare le figure relative a tutte le finestre definite in precedenza si devono usare le istruzioni: VIEW senza parametri (che definisce l'intero video come finestra) e poi CLS, poiché la sola istruzione CLS cancella il contenuto della finestra corrente.

Esempi

Si faccia riferimento all'istruzione WINDOW.

VIEW PRINT Istruzione

Definisce una finestra di testo.

VIEW PRINT [*line1* **TO** *line2*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
line1	È la linea iniziale della finestra di testo.
line2	È la linea finale della finestra di testo.

Caratteristiche

Tra le istruzioni e le funzioni che operano all'interno della finestra di testo sono incluse le funzioni CLS, LOCATE e SCREEN. Screen Editor limita le funzioni come lo scorrimento del testo e il movimento del cursore all'interno della finestra di testo.

Se non vi è specificato alcun parametro, VIEW PRINT inizializza l'intero video come finestra di testo.

Esempio

VIEW PRINT 1 TO 5

crea una finestra di testo di 5 linee all'inizio del video.

WAIT Istruzione

Sospende l'esecuzione del programma mentre viene controllato lo stato di una porta di input del sistema.

L'istruzione WAIT deve essere usata solo all'interno di un programma.

WAIT *port,i*[,*j*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
port	Rappresenta il numero di una porta compreso tra 0 e 65535.
i,j	Sono espressioni intere il cui risultato è compreso tra 0 e 255.

Caratteristiche

L'istruzione WAIT provoca la sospensione dell'esecuzione del programma finché su una porta di input del sistema non si presenta una ben specificata configurazione di bit. Viene effettuata un'operazione XOR tra il dato letto sulla porta e l'espressione intera *j*, poi AND tra il risultato e l'espressione *i*. Se il risultato è zero, GW-BASIC continua a riciclare sulle letture del valore presente sulla porta. Se il risultato è diverso da zero, l'esecuzione del programma viene ripresa dall'istruzione successiva a WAIT. Se *j* viene omesso, si assume zero come valore di default.

Usando l'istruzione WAIT può succedere di rimanere in un ciclo infinito.

Esempio

```
100 WAIT 32,2
```

WHILE...WEND Istruzione

Permette di riciclare su una serie di istruzioni fino a che una data condizione rimane vera.

WHILE *condition*
.
.
[*loop statements*]
.
.
WEND

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
condition	Può essere un'espressione di tipo logico, di confronto o numerica. GW-BASIC determina se la condizione è vera o falsa controllando il risultato dell'espressione.

ELEMENTO DI SINTASSI	SIGNIFICATO
	Un risultato diverso da zero significa vero, mentre un risultato uguale a zero significa falso. Da ciò deriva che è possibile controllare se il valore di una variabile è zero o diverso da zero semplicemente specificando il nome della variabile come condizione.

Caratteristiche

Se *condition* è diverso da zero vengono eseguite le istruzioni del ciclo fino a che non viene incontrata un'istruzione WEND. GW-BASIC a questo punto ritorna all'istruzione WHILE e controlla *condition*. Se è ancora diverso da zero, il processo viene ripetuto, se è uguale a zero, l'esecuzione riprende dall'istruzione immediatamente seguente l'istruzione WEND.

I cicli WHILE/WEND possono essere nidificati; quindi WEND deve essere associato all'istruzione WHILE fisicamente più adiacente. WHILE senza WEND provoca l'errore "WHILE without WEND", mentre l'istruzione WEND non associata a nessuna WHILE provoca l'errore "WEND without WHILE".

Esempio

```
90 'BUBBLE SORT ARRAY A$
100 FLIPS=1 'FORZA UN'ESECUZIONE DEL CICLO
110 WHILE FLIPS
```

```
115        FLIPS=0
120        FOR I=1 TO J-1
130                IF A$(I)>(I+1) THEN
                       SWAP A$(I),A$(I+1):FLIPS=1
140        NEXT I
150 WEND
```

WIDTH Istruzione

Definisce l'ampiezza della linea in caratteri. GW-BASIC aggiunge **CR** dopo l'output del numero di caratteri specificato.

In grafica definisce una finestra di testo.

Sintassi 1

WIDTH [**LPRINT**] *size*

Sintassi 2

WIDTH *#filenum,size*

Sintassi 3

WIDTH *device,size*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
size	È un'espressione intera il cui valore è compreso tra 0 e 255 e specifica la nuova ampiezza della linea.
filenum	È il numero con cui il file è stato aperto.
device	È un'espressione di tipo stringa che indica il device che deve essere usato. Valori ammessi: SCRN:, LPT1:, LPT2:, LPT3:, COM1:, COM2:,

WIDTH LPRINT *size*

Definisce l'ampiezza della linea per la stampante.

WIDTH *size* oppure WIDTH "SCRN:", *size*

Definisce l'ampiezza del video (nel Modo Testo) o seleziona una finestra di testo o cambia modo (in modo grafica). Cambiando l'ampiezza della finestra o del video, o cambiando il modo, viene effettuata la cancellazione del video.

In Modo Testo, *size* può solo avere i valori 40 o 80, selezionando così un'ampiezza video di 40 o di 80 colonne.

In uno dei modi di grafica, è possibile cambiare modo o selezionare una finestra di testo a partire dal margine sinistro del video, di ampiezza minore o uguale a 40 (Modo a Media Risoluzione), oppure di ampiezza minore uguale a 80 (Modo ad Alta o Altissima Risoluzione). L'altezza della finestra è fissa e corrisponde sempre all'altezza del video.

L'ampiezza della visualizzazione dei tasti funzione corrisponde all'ampiezza selezionata. Se il numero delle colonne visualizzate è minore di 80 si può impostare **CTRL T** allo scopo di spostare orizzontalmente la visualizzazione dei tasti funzione.

Di seguito sono riportati tutti i casi possibili.

SE il modo è...	**E l'ampiezza è...**	**ALLORA...**
0 (Testo)	40	viene selezionato un video a 40 colonne.
	80	viene selezionato un video a 80 colonne.
1 (Media Ris.)	80	il sistema viene posto nel Modo ad Alta Risoluzione (modo 2).
	8< =*size*< =40	viene creata una finestra di testo di ampiezza *size*.
2 (Alta Ris.)	40	il sistema viene posto nel Modo a Media Risoluzione (modo 1), attribuendo a *burst* il valore assegnatogli nel momento in cui sono stati usati per l'ultima volta il Modo testo o il Modo a Media Risoluzione.
	8< =*size*< =39 oppure 41< =*size*< =80	viene creata una finestra di testo di ampiezza *size*.

SE il modo è...	E l'ampiezza è...	ALLORA...
	4	viene creata un finestra di testo di ampiezza 40.
3-255 (Altissima Risoluzione)	8< =*size*< =80	viene creata una finestra di testo di ampiezza *size*.
	8< =*size*-80< =80	viene creata una finestra di testo di ampiezza *size*.

WIDTH ***#filenum*, *size***

Se il file è aperto, l'ampiezza è immediatamente sostituita con il nuovo valore. Questo permette di cambiare l'ampiezza mentre il file è aperto.

WIDTH ***device*, *size***

L'ampiezza di default della linea, per il device specificato, è posta uguale a *size*. Le ampiezze di linea dei file aperti non sono modificate.

Se il device è già aperto, la nuova ampiezza è memorizzata senza cambiare l'ampiezza corrente.

Una successiva OPEN *device* FOR OUTPUT AS # *n* usa il valore inizialmente specificato per le ampiezze.

Osservazioni

Quando l'istruzione WIDTH causa un cambiamento nel modo video, i colori sono posizionati ai loro valori di default.

È necessario disattivare la visualizzazione dei tasti funzione (usando l'istruzione KEY OFF) mentre si cambia l'ampiezza della finestra, altrimenti, se la finestra viene diminuita parte della precedente visualizzazione dei tasti funzione può rimanere sul video.

Se *size* viene posta uguale a 255, si assume che l'ampiezza della linea sia infinita; cioé GW-BASIC non inserisce mai **CR**. Pertanto, la posizione del cursore o della testina di stampa, definita dalle funzioni POS o LPOS, ritorna a 0 dopo aver superato la posizione 255. Il valore di default per i file di comunicazione è 255.

Se viene alterata l'ampiezza per un file di comunicazione, non viene modificato il buffer di trasmissione o di ricezione: GW-BASIC inserirà **CR** dopo un numero di caratteri uguali a *size* sia in trasmissione che in ricezione.

Esempio 1

```
10 PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
RUN
ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ok
WIDTH 18
Ok
RUN
ABCDEFGHIJKLMNOPQR
STUVWXYZ
OK
```

Esempio 2

```
10 WIDTH "LPT1:",5
20 OPEN "LPT1:"FOR OUTPUT AS#1
30 PRINT # 1, "1234567890"
35 PRINT # 1,""
40 WIDTH # 1,6
50 PRINT # 1, "1234567890"
RUN
```

sulla stampante si ottiene:

12345
67890

123456
7890

Esempio 3

SCREEN 1,0

viene selezionato il Modo Grafica a Media Risoluzione a colori.

WIDTH 80

il video viene posto nel Modo Grafica ad Alta Risoluzione.

WIDTH 40

il video viene riposizionato nel Modo a Media Risoluzione.

SCREEN 0,1

il video viene posto nel Modo Testo a colori 40 x 25.

WIDTH 80

il video viene posto nel Modo Testo a colori 80 x 25.

Errori Possibili

Se *size* è fuori dall'intervallo specificato, viene visualizzato l'errore "Illegal function call". Il precedente valore rimane invariato.

WINDOW Istruzione

Definisce la dimensione della finestra logica (solo modo grafica).

WINDOW [[**SCREEN**](*wx1* , *wy1*)-(*wx2* , *wy2*)]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
(*wx1*, *wy1*) (*wx2*, *wy2*)	(*wx1*, wy1) rappresentano le coordinate inferiori di sinistra e (*wx2*, *wy2*) le coordinate superiori di destra della finestra logica. L'opzione SCREEN inverte l'asse *y*, così che (*wx1*, wy1) rappresentano le coordinate superiori di sinistra, e (*wx2*, *wy2*) le coordinate inferiori di destra della finestra logica.

Caratteristiche

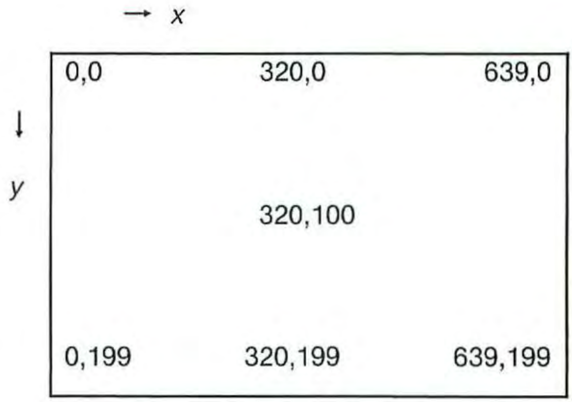
WINDOW permette di visualizzare figure in uno spazio non delimitato dalle dimensioni fisiche del video, tramite l'uso di coordinate definite dall'utente, dette coordinate logiche. Le coordinate logiche vengono convertite in coordinate fisiche automaticamente.

Se l'istruzione WINDOW non ha parametri le coordinate logiche coincidono con le coordinate fisiche.

Se vengono impostate le istruzioni:

NEW
SCREEN 2

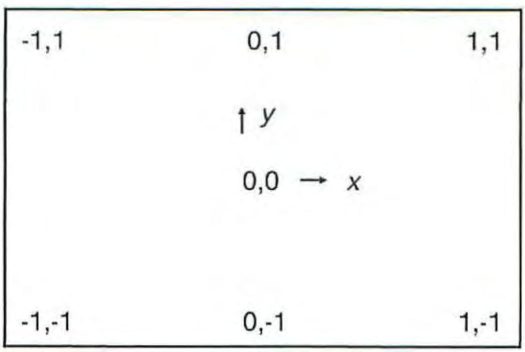
lo schermo compare così:



Impostando ora:

WINDOW (-1,-1)-(1,1)

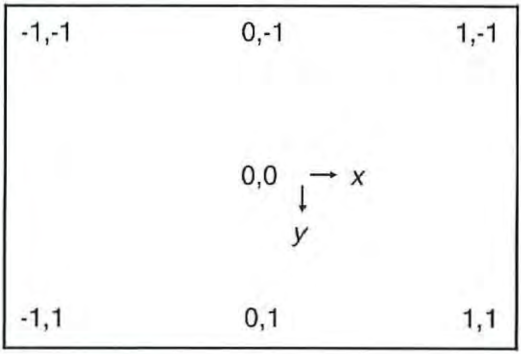
si ha:



Se viene eseguita l'istruzione:

WINDOW SCREEN (-1,-1)-(1,1)

si ha:



Orientamento degli Assi

Se l'ultima istruzione WINDOW eseguita è stata impostata con l'opzione SCREEN, gli assi X e Y della finestra logica hanno lo stesso orientamento di quelli della finestra fisica (asse X verso destra e asse Y verso il basso); se tale istruzione è stata impostata senza l'opzione SCREEN l'asse Y della finestra logica è orientato verso l'alto. Quindi, se viene realizzato lo stesso disegno due volte utilizzando prima WINDOW e poi WINDOW SCREEN, la seconda immagine viene visualizzata sulla finestra fisica in modo simmetrico rispetto all'asse X.

L'esempio seguente visualizza un triangolo con il vertice verso il basso, alla prima esecuzione delle istruzioni alla linea 1000, poi visualizza lo stesso triangolo con il vertice verso l'alto, alla seconda esecuzione delle istruzioni alla linea 1000.

```
10 SCREEN 1
20 VIEW
30 CLS
40 INPUT X1,Y1,X2,Y2
50 VIEW (X1,Y1)-(X2,Y2),,1
60 WINDOW (-1000,-1000)-(1000,1000)
70 GOSUB 1000
80 WINDOW SCREEN (-1000,-1000)-(1000,1000):GOSUB 1000
90 END
1000 LINE (-500,500)-(500,500):LINE -(0, -500): LINE -(-500,500):
     A$ = INPUT$(1):CLS:RETURN
```

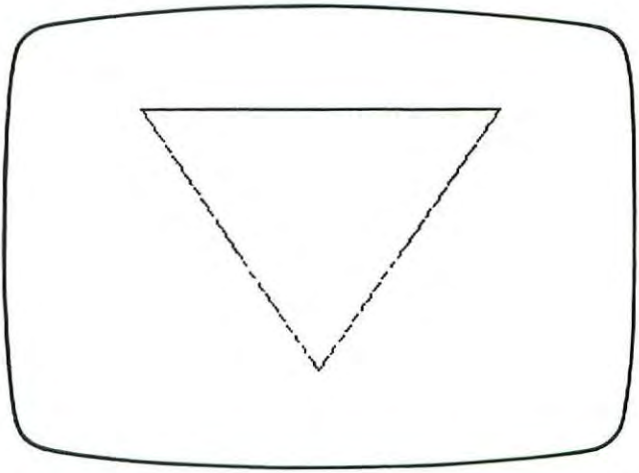


Fig. 8-5 Esempio di Orientamento Assi (cont.)

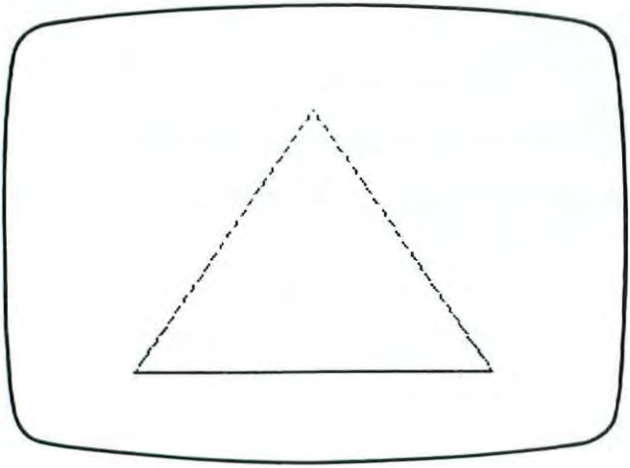


Fig. 8-5 Esempio di Orientamento Assi

Proiezione dell'Immagine su Video

Tutti i punti della figura realizzata, con istruzioni di grafica, sulla finestra logica (definita con WINDOW) vengono proiettati sui punti della finestra fisica definita con l'istruzione VIEW. Se tale istruzione viene impostata con l'opzione SCREEN, i punti della figura realizzata sulla finestra logica vengono proiettati sui punti dell'intero video, ma solo i punti interni alla finestra fisica corrente sono illuminati (effetto clipping).

Esempio 1

```
10 PI = 3.14159
20 SCREEN 3:CLS:KEY OFF
30 WINDOW (0,-1)-(2*PI,1)
40 VIEW SCREEN (1 ,1 )-(200,200),,1
50 GOSUB 1000
```

```
60 VIEW SCREEN (440,301)-(638,398),,1
70 GOSUB 1000
80 VIEW SCREEN (1,201)-(638,300),,1
90 GOSUB 1000
100 END
1000 PSET (0,0)
1010 FOR ALPHA=0 TO 2*PI STEP PI/100
1020 LINE -(ALPHA,SIN(ALPHA))
1030 NEXT
1040 A$=INPUT$(1)
1050 RETURN
```

Viene tracciata una sinusoide ed evidenziato l'effetto clipping, definendo una finestra logica (linea 30) è piú finestre fisiche tramite l'istruzione VIEW SCREEN (linee 40, 60 e 80).

L'istruzione alla linea 1040 permette di sospendere l'esecuzione del programma fino alla pressione di un qualsiasi tasto.

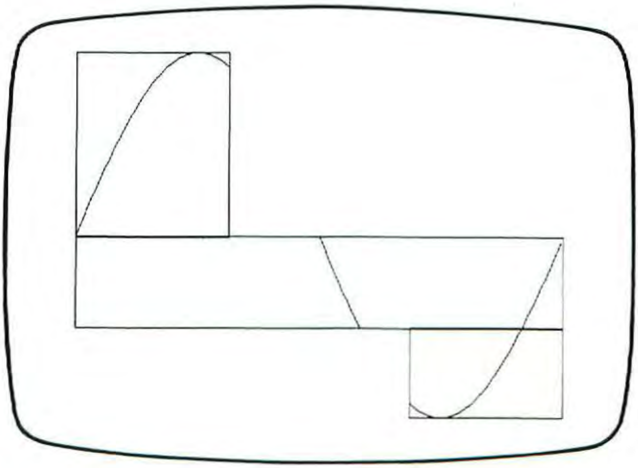


Fig. 8-6 Esempio 1

Variando l'area di una delle due finestre (logica o fisica) cambia la corrispondenza tra i loro punti e si possono ottenere immagini diverse dello stesso disegno.

Variando l'area di una finestra e mantenendo costante il rapporto tra i lati si ottengono immagini simili tra loro; variando questo rapporto si ottengono immagini distorte.

Esempio 2

```
5 CLS
10 SCREEN 1
20 INPUT X1,Y1,X2,Y2
25 VIEW
30 VIEW (X1,Y1)-(X2,Y2),,1
40 WINDOW (-100,-100)-(100,100)
50 GOSUB 1000
60 A$ = INPUT$(1):CLS:GOTO 20
1000 INPUT R
1010 PSET (R,0)
1020 PI = 3.14159
1030 FOR ALFA = 0 TO 2*PI STEP PI/100
1040 LINE -(R*COS(ALFA),R*SIN(ALFA))
1050 NEXT:RETURN
```

Viene tracciato un cerchio, come un poligono con un gran numero di lati, tramite l'istruzione LINE e un ciclo FOR/NEXT. Lo stesso cerchio tracciato con l'istruzione CIRCLE non avrebbe subito distorsioni, variando il rapporto tra i lati della finestra fisica.

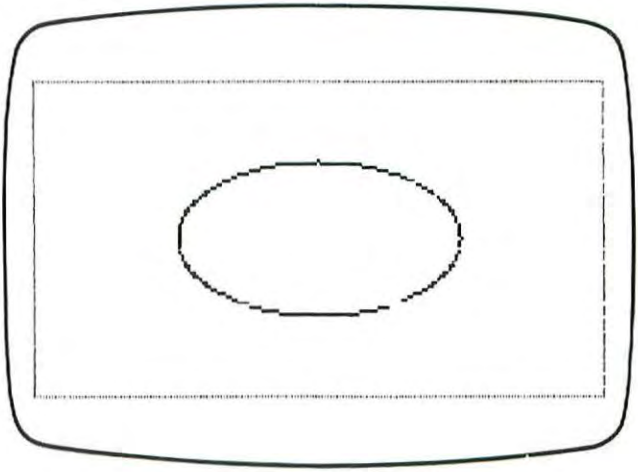


Fig. 8-7 Esempio 2

Lasciando invariata la finestra fisica e variando l'area della finestra logica (tramite WINDOW) è possibile aumentare o ridurre la dimensione dell'immagine oppure evidenziare un particolare.

Esempio 3

```
10 SCREEN 3
20 VIEW
30 CLS
40 VIEW (1,100)-(638,300),,1
50 WINDOW (-100,-100)-(100,100)
60 GOSUB 1000
70 A$=INPUT$(1)
80 CLS
90 WINDOW (-40,-40)-(40,40)
100 GOSUB 1000
110 END
1000 LINE (-50,-30)-(50,30)
1010 CIRCLE (0,0),20
1020 RETURN
```

Viene tracciato un cerchio tagliato da un segmento di retta ed evidenziato l'effetto zoom, definendo due diverse finestre logiche.

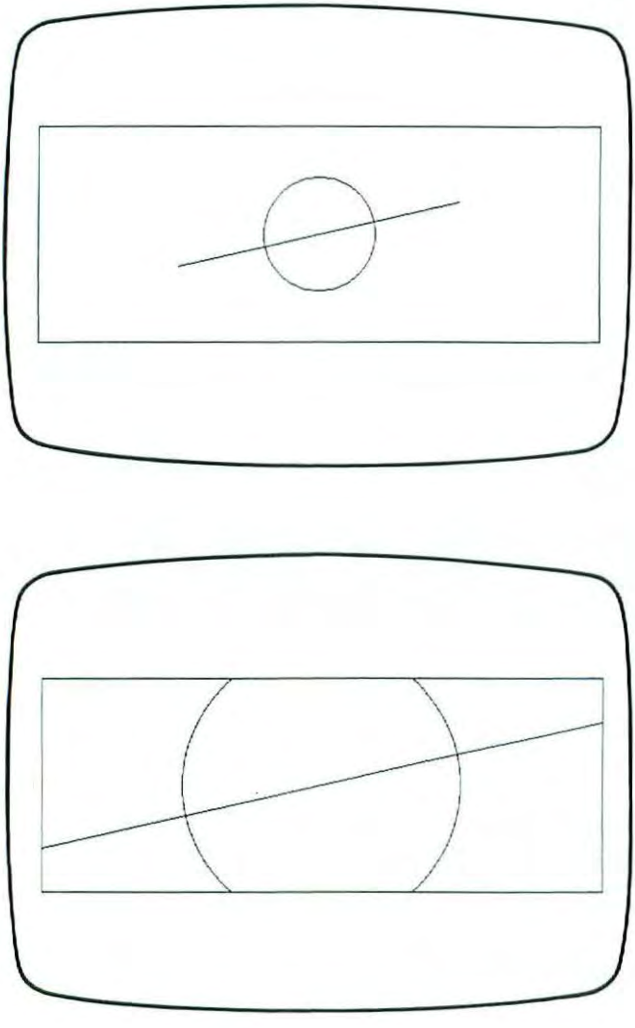


Fig. 8-8 Esempio 3

È possibile anche aumentare o ridurre la dimensione dell'immagine lasciando invariata la finestra logica e variando l'area della finestra fisica (tramite VIEW).

Esempio 4

```
10 SCREEN 3:KEY OFF:CLS
20 PI = 3.14159
30 VIEW
40 CLS
50 WINDOW (-100,-100)-(100,100)
60 VIEW (200,1)-(400,200),,1
70 CIRCLE (0,0),100
80 PAINT (30,30)
90 VIEW (300,201)-(400,398),,1
100 CIRCLE (0,0),100
110 PAINT (30,30)
```

Questo esempio traccia un cerchio e lo colora evidenziando la possibilità di definire piú finestre fisiche e modificare l'immagine su video variando le dimensioni di tali finestre. Vengono infatti definite in successione due finestre fisiche (linee 60 e 90).

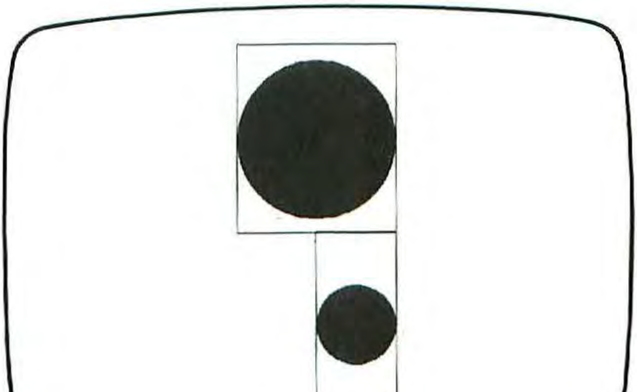


Fig. 8-9 Esempio 4

Si noti che non si ha effetto clipping poiché non viene modificata la finestra logica e non è usata l'opzione SCREEN per VIEW. Inoltre il cerchio tracciato resta sempre tale poiché è stata usata l'istruzione CIRCLE.

WRITE Istruzione

Scrive dati su video.

WRITE [*list-of-expressions*]

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
list-of-expressions	È una lista di espressioni numeriche e/o stringa che devono essere separate da virgole.

Caratteristiche

I valori delle espressioni vengono visualizzati. Se non viene indicata nessuna espressione, viene visualizzata una linea vuota.

I valori devono essere separati tra loro da virgole. Le stringhe vengono delimitate da doppi apici. I valori numerici vengono visualizzati usando lo stesso formato dell'istruzione PRINT, ma non sono seguiti da spazi. Dopo l'ultimo elemento della lista GW-BASIC inserisce **CR LF**.

Esempio

```
10 A=80:B=90:C$="THAT'S ALL"
20 WRITE A,B,C$
RUN
 80,90,"THAT'S ALL"
Ok
```

WRITE# Istruzione

Scrive dati su un file di tipo sequenziale.

WRITE *#filenum,list-of-expressions*

Dove

ELEMENTO DI SINTASSI	SIGNIFICATO
filenum	È il numero con cui il file è stato aperto in modo O.
list-of-expressions	Lista di espressioni di tipo numerico o stringa. Devono essere separate da virgole.

Caratteristiche

La differenza tra le istruzioni WRITE# e PRINT# è costituita dall'inserimento da parte di WRITE# di virgole tra i campi, mentre questi vengono scritti sul file, e dall'uso di doppi apici per delimitare le stringhe. Pertanto, non è necessario che l'utente inserisca esplicitamente delimitatori nella lista. Una sequenza di **CR LF** viene inserita dopo la registrazione dell'ultimo campo della lista nel file.

Esempio

```
10 A$="CAMERA":B$="93604-1"
20 WRITE #1,A$,B$
```

L'istruzione 20 scrive le seguenti informazioni su disco:

```
"CAMERA","93604-1"
```

Una successiva istruzione INPUT#, del tipo:

```
30 INPUT#1,A$,B$
```

assegna "CAMERA" a A$ e "93604-1" a B$.

A. CODICI E MESSAGGI DI ERRORE

SOMMARIO

Questa appendice descrive tutti i codici e i messaggi di errore emessi da GW-BASIC.

NUMERO	MESSAGGIO
1	NEXT without FOR È stata riscontrata un'istruzione NEXT senza una corrispondente istruzione FOR.
2	Syntax error È stata riscontrata una linea che contiene una sequenza di caratteri non corretta (parole chiave scritte in modo errato, parentesi non richiuse, punteggiatura non corretta, ecc.). GW-BASIC entra automaticamente in modo edit sulla linea che ha causato l'errore.
3	RETURN without GOSUB È stata riscontrata una istruzione RETURN senza una corrispondente istruzione GOSUB.
4	Out of DATA È stata eseguita un'istruzione READ ma nel programma non ci sono più istruzioni DATA con dati da leggere.
5	Illegal function call Ad una funzione numerica o stringa è stato passato un parametro fuori dai valori specificati. Un errore di questo tipo (FC) si può anche verificare a causa di:

NUMERO	MESSAGGIO
	• indice di una matrice negativo o troppo grande • funzione LOG con argomento negativo o uguale a zero • funzione SQR con argomento negativo • mantissa negativa con esponente non intero • chiamata a una funzione URS a cui non è ancora stato assegnato l'indirizzo iniziale • assegnazione di un argomento non corretto a MID$, LEFT$, RIGT$, INP, OUT, WAIT, PEEK, POKE, TAB, SPC, STRING$, SPACE$, INSTR oppure ON...GOTO • numero di record negativo usato con le istruzioni GET (File) o PUT (File).
6	Overflow Il risultato di un calcolo è troppo grande per essere rappresentato nel formato dei numeri GW-BASIC. Se si verifica un underflow, il risultato è uguale a zero e l'esecuzione continua senza errori.
7	Out of memory Un programma è troppo vasto, ha troppi cicli, troppi sottoprogrammi, troppe variabili oppure ha espressioni troppo complesse da calcolare.

NUMERO	MESSAGGIO
8	Undefined line number Un'istruzione GOTO, GOSUB, IF...THEN...ELSE o DELETE fa riferimento a una linea inesistente.
9	Subscript out of range È stato indicato un elemento di vettore con un indice fuori dalle dimensioni del vettore o con un numero sbagliato di indici.
10	Duplicate Definition Due istruzioni DIM fanno riferimento allo stesso vettore oppure viene associata un'istruzione DIM ad un vettore, la cui dimensione è già stata fissata pari al valore di default 10, oppure viene eseguita un'istruzione OPTION BASE dopo che il vettore è stato dimensionato.
11	Division by zero È stata riscontrata, in un'espressione, una divisione per zero oppure il valore zero è stato elevato a una potenza negativa. Nel primo caso il risultato è l'infinito di macchina (con il segno appropriato); nel secondo caso il risultato è l'indefinito di macchina con segno positivo.
12	Illegal direct È stata introdotta un'istruzione in modo immediato per la quale non è permessa tale introduzione.

NUMERO	MESSAGGIO
13	Type mismatch Viene assegnato un valore numerico a una variabile stringa o viceversa; ad una funzione che richiede un argomento numerico è stato assegnato un argomento stringa o viceversa.
14	Out of string space Variabili stringa hanno provocato il superamento della memoria utente disponibile. GW-BASIC alloca per le stringhe lo spazio in modo dinamico, finché non si verifica un errore di questo tipo.
15	String too long Si è tentato di creare una stringa con più di 255 caratteri.
16	String formula too complex L'espressione stringa è troppo lunga o troppo complessa, è opportuno dividerla in più espressioni semplici.
17	Can't continue Si è cercato di riprendere l'esecuzione di un programma che: • si è fermato a causa di un errore

▶

NUMERO	MESSAGGIO
	• è stato modificato durante un'interruzione dell'esecuzione • non esiste.
18	Undefined user function Viene richiamata una funzione prima della corrispondente definizione (tramite l'istruzione DEF).
19	No RESUME Il controllo è stato passato a una routine di gestione degli errori che non contiene un'istruzione RESUME.
20	RESUME without error È stata riscontrata un'istruzione RESUME prima di entrare in una routine di gestione degli errori.
22	Missing operand Un'espressione contiene un operatore non seguito da operando.
23	Line buffer overflow Si è tentato di introdurre una linea formata da troppi caratteri.

NUMERO	MESSAGGIO
24	Device Timeout GW-BASIC non ha ricevuto le informazioni da un device di I/O entro un intervallo di tempo predeterminato.
25	Device Fault È stato introdotto un identificatore di device errato.
26	FOR without NEXT È stata riscontrata un'istruzione FOR senza una corrispondente istruzione NEXT.
27	Out of Paper La stampante è senza carta oppure è spenta. Inserire la carta, assicurarsi che la stampante sia accesa e continuare.
29	WHILE without WEND Un'istruzione WHILE non ha una corrispondente istruzione WEND.
30	WEND without WHILE Un'istruzione WEND non ha una corrispondente istruzione WHILE.

NUMERO	MESSAGGIO
50	FIELD overflow Un'istruzione FIELD cerca di allocare più byte di quelli specificati come lunghezza di record in un file ad accesso diretto.
51	Internal error Si è verificata un'anomalia interna a GW-BASIC.
52	Bad file number Un'istruzione o un comando fanno riferimento a un file il cui numero non appartiene all'intervallo specificato all'inizializzazione oppure il file corrispondente non è aperto.
53	File not found Un comando LOAD, KILL o NAME o un'istruzione OPEN fanno riferimento a un file che non esiste sul disco corrente.
54	Bad file mode Si è tentato di usare istruzioni PUT (File), GET (File) di caricare con LOAD un file ad accesso diretto o di

NUMERO	MESSAGGIO
	eseguire un'istruzione OPEN con un metodo di accesso diverso da I, O, A o R.
55	File already open È stata eseguita un'istruzione OPEN in modo output sequenziale su un file già aperto oppure è stato emesso un comando KILL su un file già aperto.
57	Device I/O Error Si è verificato un errore di I/O durante un'operazione su un device periferico. È un errore non recuperabile.
58	File already exists Il nome di file specificato in un comando NAME è identico a uno che esiste sul disco selezionato.
61	Disk full Tutto lo spazio disponibile su disco è stato usato.
62	Input past end È stata eseguita un'istruzione INPUT su un file i cui dati erano già stati letti oppure su un file vuoto. Per evitare questo errore si deve usare la funzione EOF per accertarsi della fine del file.

NUMERO	MESSAGGIO
63	Bad record number Il numero di record usato in un'istruzione PUT (File) o GET (File) è esterno all'intervallo consentito, cioé è uguale a zero oppure maggiore di 16777215.
64	Bad file name In un comando LOAD, SAVE o KILL o in un'istruzione OPEN è stato usato un nome di file non valido (per esempio con troppi caratteri).
66	Direct statement in file Durante il caricamento di un file in formato ASCII è stata incontrata un'istruzione diretta. L'operazione di caricamento (tramite LOAD) viene terminata.
67	Too many files Si è tentato di creare un nuovo file (con il comando SAVE o l'istruzione OPEN) quando la directory era già satura.
68	Device Unavailable Si è tentato di aprire un file su un device inesistente. Per esempio, non è presente l'hardware per supportare il device, come LPT2: o LPT3, oppure è stato disabilitato dall'utente. Ciò avviene se viene eseguita un'istruzione OPEN "COM1:... dopo che è stato disabilitato il supporto RS232 tramite lo switch /C:0 impostando il comando GWBASIC.

NUMERO	MESSAGGIO
69	Communication buffer overflow Non è stato riservato abbastanza spazio per un'istruzione di input di comunicazione e la coda di input risulta piena. In questo caso occorre usare un'istruzione di tipo ON ERROR GOTO per ritentare l'introduzione. Successive introduzioni tenteranno di rimuovere questo errore, se i caratteri non vengono ricevuti ad una velocità più alta dell'elaborazione del programma. Sono disponibili diverse opzioni: • Aumentare la misura del buffer COM tramite lo switch /C: • Realizzare un protocollo "hand-shaking" con host/satellite regolati da XON/XOFF (vedere l'esempio di programmazione TTY in cui la trasmissione viene interrotta il tempo necessario a evitare l'overflow) • Usare una frequenza (baud rate) di trasmissione e ricezione più bassa
70	Permission Denied Si verifica quando viene fatto un tentativo di scrivere su un disco che è protetto da scrittura. Si deve usare un'istruzione ON ERROR GOTO per scoprire questa situazione e definire l'azione da intraprendere. Altri possibili errori di hard-disk sono: 71, 72 e 74. Questo errore si verifica anche con le istruzioni LOCK, UNLOCK e OPEN, se l'accesso ad un file è limitato o se MS-NET non è stato installato.

▶

NUMERO	MESSAGGIO
71	Disk not ready Si verifica quando lo sportello del drive di disco è aperto o quando il disco non è sul drive. Per superare l'errore usare anche in questo caso l'istruzione ON ERROR GOTO.
72	Disk media error Si verifica quando viene rilevata una anomalia hardware o sul supporto. Di solito indica supporto danneggiato; si devono copiare tutti i file esistenti su un nuovo disco e riformattare il disco danneggiato. FORMAT segnala le tracce danneggiate e le pone in una condizione di "badtrack". La parte del disco rimanente è di nuovo utilizzabile.
74	Rename across disks Si è tentato di assegnare a un file un nuovo nome, che è stato dichiarato su un disco, diverso da quello specificato per il nome precedente.
75	Path/File Access Error MS-DOS non ha potuto effettuare una connessione corretta tra pathname e filename durante un'operazione di OPEN, MKDIR, CHDIR o RMDIR. L'operazione non viene completata.

NUMERO	MESSAGGIO
76	Path not found MS-DOS non ha trovato il pathname specificato durante un'operazione di OPEN, MKDR, CHDIR o RMDIR. L'operazione non viene completata.
**	Can't continue after SHELL Non c'è un numero d'errore. SHELL scopre al rientro da un processo child che non c'è sufficiente spazio in memoria per permettere a GW-BASIC di continuare. Vengono chiusi tutti i file aperti e si ritorna a MS-DOS.

B. FUNZIONI MATEMATICHE

SOMMARIO

Questa appendice descrive come calcolare funzioni matematiche di uso frequente che non sono intrinseche a GW-BASIC.

FUNZIONI MATEMATICHE

Di seguito è indicato come è possibile esprimere alcune funzioni matematiche di uso frequente utilizzando quelle intrinseche a GW-BASIC.

FUNZIONE	EQUIVALENTE GW-BASIC
SECANTE	SEC(x) = 1/COS(x) quando x< >1.570796
COSECANTE	CSC(x) = 1/SIN(x) quando x< >0
COTAGENTE	COT(x) = 1/TAN(x) quando x< >0)
INVERSA DEL SENO	ARCSIN(x) = ATN(x/SQR(1-x*x))
INVERSA DEL COSENO	ARCCOS(x) = 1.570796-ATN(x/SQR(1-x*x)) quando ABS(x)<1
INVERSA DELLA SECANTE	ARCSEN(x) = ATN(SQR(x*x-1) +SGN(SGN(x)-1)*1.570796 quando ABS(x)> = 1
INVERSA DELLA COSECANTE	ARCCSC(x) = ATN(1/SQR(x*x-1)) +(SGN(x)-1)*1.570796 quando ABS(x)>1
INVERSA DELLA COTANGENTE	ARCCOT(x) = 1.570796-ATN(x)
SENO IPERBOLICO	SINH(x) = (EXP(x)-EXP(-x))/2
COSENO IPERBOLICO	COSH(x) = (EXP(x) + EXP(-x))/2
TANGENTE IPERBOLICA	TANH(x) = (EXP(x)-EXP(-x))/ (EXP(x) + EXP(-x))
SECANTE IPERBOLICA	SECH(x) = 2/(EXP(x) + EXP(-x))

FUNZIONE	EQUIVALENTE GW-BASIC
COSECANTE IPERBOLICA	CSCH(x) = 2/(EXP(x)-EXP(-x)) quando x< >0
COTANGENTE IPERBOLICA	COTH(x) = EXP(x) + EXP(-x))/ (EXP(x)-EXP(-x)) quando x< >0
INVERSA DEL SENO IPERBOLICA	ARCSINH(x) = LOG(x + SQR(x * x + 1)
INVERSA DEL COSENO IPERBOLICO	ARCCOSH(x) = LOG(x + SQR(x*x-1) quando x> = 1
INVERSA DELLA TAGENTE IPERBOLICA	ARCTANH(x) = LOG((1 + x)/(1-x))/2 quando ABS(x)<1
INVERSA DELLA SECANTE IPERBOLICA	ARCSECH(x) = LOG((SQR(1-x*x) + 1)/x) quando 0<x< = 1
INVERSA DELLA COSECANTE IPERBOLICA	ARCCSH(x) = LOG((SGN(x)*SQR(x*x + 1) + 1)/x) quando x>0
INVERSA DELLA COTANGENTE IPERBOLICA	ARCCOTH(x) = LOG((x + 1)/(x-1))/2 quando ABS(x)>1
LOGARITMO IN BASE 'a'	LOGA(x) = LOG(x)/LOG(a) quando a>0 e x>0

È possibile definire una funzione matematica derivata tramite l'istruzione DEF FN, per evitare di impostare la formula ogni volta che viene utilizzata nel programma.

Sia 'x' che 'a' possono essere costanti numeriche, variabili numeriche, elementi di matrice, funzioni o espressioni. I valori di 'x' o di 'a' che possono provocare messaggi di errore, vengono indicati.

C. CODICI DEI CARATTERI ASCII

SOMMARIO

Questa appendice riporta la tabella dei caratteri ASCII.

CODICI DEI CARATTERI ASCII

Questa tabella illustra i codici decimali, esadecimali e i simboli visualizzabili dei 256 elementi appartenenti al set standard dei caratteri ASCII.

DEC	HEX	CHARACTER	DEC	HEX	CHARACTER	DEC	HEX	CHARACTER	DEC	HEX	CHARACTER
000	00	BLANK (NULL) (NULL)	016	10	► (DLE)	032	20	BLANK (SPACE)	048	30	0
001	01	☺ (SOH)	017	11	◄ (DC1)	033	21	!	049	31	1
002	02	☻ (STX)	018	12	↕ (DC2)	034	22	"	050	32	2
003	03	♥ (ETX)	019	13	‼ (DC3)	035	23	#	051	33	3
004	04	♦ (EOT)	020	14	¶ (DC4)	036	24	$	052	34	4
005	05	♣ (ENQ)	021	15	§ (NAC)	037	25	%	053	35	5
006	06	♠ (ACK)	022	16	▬ (SYN)	038	26	&	054	36	6
007	07	• (BEL)	023	17	↨ (ETB)	039	27	'	055	37	7
008	08	◘ (BS)	024	18	↑ (CAN)	040	28	(	056	38	8
009	09	○ (HT)	025	19	↓ (EM)	041	29	)	057	39	9
010	0A	◙ (LF)	026	1A	→ (SUB)	042	2A	*	058	3A	:
011	0B	♂ (VT)	027	1B	← (ESC)	043	2B	+	059	3B	;
012	0C	♀ (FF)	028	1C	∟ (FS)	044	2C	,	060	3C	<
013	0D	♪ (CR)	029	1D	↔ (GS)	045	2D	—	061	3D	=
014	0E	♫ (SO)	030	1E	▲ (RS)	046	2E	.	062	3E	>
015	0F	☼ (SI)	031	1F	▼ (US)	047	2F	/	063	3F	?

Tab. C-1 Set Standard dei Caratteri ASCII (cont.)

DEC	HEX	CHARACTER	DEC	HEX	CHARACTER	DEC	HEX	CHARACTER	DEC	HEX	CHARACTER
064	40	@	080	50	P	096	60	‘	112	70	p
065	41	A	081	51	Q	097	61	a	113	71	q
066	42	B	082	52	R	098	62	b	114	72	r
067	43	C	083	53	S	099	63	c	115	73	s
068	44	D	084	54	T	100	64	d	116	74	t
069	45	E	085	55	U	101	65	e	117	75	u
070	46	F	086	56	V	102	66	f	118	76	v
071	47	G	087	57	W	103	67	g	119	77	w
072	48	H	088	58	X	104	68	h	120	78	x
073	49	I	089	59	Y	105	69	i	121	79	y
074	4A	J	090	5A	Z	106	6A	j	122	7A	z
075	4B	K	091	5B	[	107	6B	k	123	7B	{
076	4C	L	092	5C	\	108	6C	l	124	7C	¦
077	4D	M	093	5D	]	109	6D	m	125	7D	}
078	4E	N	094	5E	∧	110	6E	n	126	7E	~
079	4F	O	095	5F	_	111	6F	o	127	7F	△

Tab. C-1 Set Standard dei Caratteri ASCII (cont.)

DEC	HEX	CHARACTER	DEC	HEX	CHARACTER	DEC	HEX	CHARACTER	DEC	HEX	CHARACTER
128	80	Ç	144	90	É	160	A0	á	176	B0	░
129	81	ü	145	91	æ	161	A1	í	177	B1	▒
130	82	é	146	92	Æ	162	A2	ó	178	B2	▓
131	83	â	147	93	ô	163	A3	ú	179	B3	│
132	84	ä	148	94	Ö	164	A4	ñ	180	B4	┤
133	85	à	149	95	ò	165	A5	Ñ	181	B5	╡
134	86	å	150	96	û	166	A6	ª	182	B6	╢
135	87	ç	151	97	ù	167	A7	º	183	B7	╖
136	88	ê	152	98	ÿ	168	A8	¿	184	B8	╕
137	89	ë	153	99	Ö	169	A9	⌐	185	B9	╣
138	8A	è	154	9A	Ü	170	AA	¬	186	BA	║
139	8B	ï	155	9B	¢	171	AB	1/2	187	BB	╗
140	8C	î	156	9C	£	172	AC	1/4	188	BC	╝
141	8D	ì	157	9D	¥	173	AD	¡	189	BD	╜
142	8E	Ä	158	9E	Pt	174	AE	«	190	BE	╛
143	8F	Å	159	9F	f	175	AF	»	191	BF	┐

Tab. C-1 Set Standard dei Caratteri ASCII (cont.)

DEC	HEX	CHARACTER	DEC	HEX	CHARACTER	DEC	HEX	CHARACTER	DEC	HEX	CHARACTER
192	C0	└	208	D0	╨	224	E0	α	240	F0	≡
193	C1	┴	209	D1	╤	225	E1	β	241	F1	±
194	C2	┬	210	D2	╥	226	E2	Γ	242	F2	≥
195	C3	├	211	D3	╙	227	E3	π	243	F3	≤
196	C4	─	212	D4	╘	228	E4	Σ	244	F4	⌠
197	C5	┼	213	D5	╒	229	E5	σ	245	F5	⌡
198	C6	╞	214	D6	╓	230	E6	µ	246	F6	÷
199	C7	╟	215	D7	╫	231	E7	τ	247	F7	≈
200	C8	╚	216	D8	╪	232	E8	φ	248	F8	°
201	C9	╔	217	D9	┘	233	E9	θ	249	F9	•
202	CA	╩	218	DA	┌	234	EA	Ω	250	FA	·
203	CB	╦	219	DB	█	235	EB	δ	251	FB	√
204	CC	╠	220	DC	▄	236	EC	∞	252	FC	n
205	CD	═	221	DD	▌	237	ED	∅	253	FD	2
206	CE	╬	222	DE	▐	238	EE	∈	254	FE	■
207	CF	╧	223	DF	▀	239	EF	∩	255	FF	BLANK 'FF'

Tab. C-1 Set Standard dei Caratteri ASCII

Per Danimarca e Norvegia alcuni caratteri differiscono dallo standard. Questi caratteri con i loro codici decimali ed esadecimali sono riportati nella tabella seguente.

DEC	HEX	CHARACTER
155	9B	ø
157	9D	Ø
158	9E	Ŀ
159	9F	ŀ
166	A6	õ
167	A7	Õ
169	A9	ã
170	AA	Ã
171	AB	ℓ
172	AC	ʼn
174	AE	³
175	AF	¤

Tab. C-2 Caratteri per Danimarca e Norvegia

D. TABELLE DI CONVERSIONE

SOMMARIO

Questa appendice riporta le tabelle di conversione da decimale a esadecimale e da decimale a binario, ottale ed esadecimale.

BYTE				BYTE			
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0
1	4096	1	256	1	16	1	1
2	8192	2	512	2	32	2	2
3	12288	3	768	3	48	3	3
4	16384	4	1024	4	64	4	4
5	20480	5	1280	5	80	5	5
6	24576	6	1536	6	96	6	6
7	28672	7	1792	7	112	7	7
8	32768	8	2048	8	128	8	8
9	36864	9	2304	9	144	9	9
A	40960	A	2560	A	160	A	10
B	45056	B	2816	B	176	B	11
C	49152	C	3072	C	192	C	12
D	53248	D	3328	D	208	D	13
E	57344	E	3584	E	224	E	14
F	61440	F	3840	F	240	F	15
4		3		2		1	

Tab. D-1 Conversione Decimale/Esadecimale

DECIMAL	BINARY	OCTAL	HEX
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	8
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Tab. D-2 Conversione Decimale/Binario/Ottale/Esadecimale

E. CONVERSIONE DI PROGRAMMI IN GW-BASIC

SOMMARIO

Questa appendice schematizza i passi da seguire per convertire i programmi, scritti in altri linguaggi BASIC, in GW-BASIC.

INDICE

INTRODUZIONE

Il linguaggio GW-BASIC è simile a molti linguaggi BASIC. Sui Personal Computer possono essere utilizzati programmi scritti per una grande varietà di altri microcomputer. Per quanto riguarda programmi scritti in linguaggi BASIC diversi da GW-BASIC, possono essere necessarie alcune piccole modifiche prima di eseguirli. Questa appendice mette in evidenza alcune aree specifiche da esaminare quando si convertono tali programmi.

COME DIMENSIONARE LE STRINGHE

LUNGHEZZA DELLE STRINGHE

Le stringhe GW-BASIC sono di lunghezza variabile: perciò devono essere cancellate tutte le istruzioni che dichiarano la lunghezza delle stringhe. Per esempio un'istruzione che dimensiona un vettore stringa per "J" elementi di lunghezza 'I' come:

DIM A$ (I,J)

deve essere convertita in questo modo:

DIM A$ (J)

SOTTOSTRINGHE

In GW-BASIC vengono usate le funzioni seguenti per estrarre sottostringhe da stringhe:

LEFT$

MID$

RIGHT$

Altre forme come: A$ (I) (per accedere all'I-mo carattere di A$) oppure A $ (I,J) (per estrarre una sottostringa di A$ dalla posizione I alla posizione J) devono essere modificate come segue:

Altri linguaggi BASIC	GW-BASIC
X$(I) = A$(I)	X$ = MID$(A$,I,1)
X$ = A$(I,J)	X$ = MID$(A$,I,J-I + 1)

Se il riferimento alla sottostringa si trova sul lato sinistro di un'assegnazione e viene usato X$ per sostituire i caratteri di A$, la conversione deve avvenire come segue:

Altri linguaggi BASIC	GW-BASIC
A$(I) = X$	MID$(A$,I,1) = X$
A$(I,J) = X$	MID$(A$,I,J-I + 1) = X$

CONCATENAMENTO

GW-BASIC usa un segno più (+) per indicare il concatenamento di stringhe. Altri linguaggi BASIC usano una virgola (,) o una congiunzione commerciale (&), che devono essere quindi sostituite.

FUNZIONI MAT

Alcuni linguaggi BASIC comprendono funzioni MAT per facilitare il calcolo matriciale. Per adattare un programma che usa queste funzioni all'ambiente GW-BASIC è necessario riscrivere il programma inserendovi cicli FOR...NEXT.

ASSEGNAZIONI MULTIPLE

Alcuni linguaggi BASIC permettono di usare la seguente sintassi:

```
10 LET D=E=0
```

per specificare che D ed E sono uguali a zero. GW-BASIC interpreta il secondo segno di uguale come un operatore logico e pone D uguale a -1, se E è uguale a 0. Questa istruzione deve quindi essere divisa in due istruzioni di assegnazione come segue:

```
10 D = 0:E = 0
```

ISTRUZIONI MULTIPLE

Le istruzioni multiple comprese in una linea GW-BASIC devono essere separate da due punti (:) mentre altri linguaggi BASIC usano una barra inversa (\).

ISTRUZIONI PEEK E POKE

L'esecuzione di programmi contenenti istruzioni PEEK o POKE può variare da macchina a macchina. È quindi necessario analizzare l'utilizzo di queste istruzioni in altri programmi BASIC, prima di tradurre le stesse funzioni in GW-BASIC.

ISTRUZIONI IF...THEN...[ELSE...]

Non tutti i linguaggi BASIC mettono in evidenza la clausola opzionale ELSE, che viene eseguita nel caso di un controllo che si rivela falso. Per esempio un'istruzione BASIC può essere in origine:

```
10 IF D=E THEN 30
20 PRINT "NOT EQUAL": GOTO 40
30 PRINT "EQUAL"
40 REM CONTINUE
```

La sequenza di istruzioni precedenti funziona correttamente, ma può essere ottimizzata in GW-BASIC come segue:

```
10 IF D = E THEN PRINT "EQUAL" ELSE PRINT "NOT EQUAL"
20 REM CONTINUE
```

OPERAZIONI DI I/O SU FILE

In alcuni linguaggi BASIC le operazioni di I/O su disco si svolgono in modo diverso da GW-BASIC. Ulteriori informazioni sui file sono riportate nel Capitolo 4.

GRAFICA

Il modo di selezionare gli attributi video e di tracciare un'immagine su video può variare da un linguaggio BASIC all'altro. Ulteriori informazioni di grafica sono riportate nel Capitolo 5.

F. GESTIONE DELLA MEMORIA

SOMMARIO

Questa appendice contiene informazioni tecniche specifiche inerenti a GW-BASIC.

Comprende l'allocazione delle variabili, la rappresentazione interna dei numeri in virgola mobile, la struttura del File Control Block e la mappa della memoria.

INDICE

ALLOCAZIONE DELLE VARIABILI

Ogni variabile definita viene associata ad un record allocato nel Segmento Dati GW-BASIC. Tale record è strutturato in piú campi, ciascuno contenente un attributo della variabile (tipo, nome, descrittore di stringa) o il suo valore. Il record associato ad una variabile numerica ne contiene anche il valore, mentre il record associato ad una variabile stringa contiene un campo (descrittore di stringa) che punta all'indirizzo, nello spazio stringhe, in cui è memorizzato il valore della variabile.

La seguente tabella illustra il modo in cui una variabile numerica o stringa viene allocata in memoria. In particolare la prima colonna indica lo spiazzamento del campo dall'indirizzo iniziale del record, la seconda colonna indica la lunghezza (in byte), la terza il contenuto e la quarta riporta ulteriori informazioni.

SPIAZZAMENTO	LUNGHEZZA	DESCRIZIONE	COMMENTI
0	1	Tipo	Il valore del tipo di variabile è: 2 (interi) 3 (strighe) 4 (numeri in semplice precisione) 8 (numeri in doppia precisione).
1	da 3 a n+3	Nome	- I byte con spiazzamento 1 e 2 contengono i primi due caratteri del nome. - Il byte con spiazzamento 3 contiene un numero che indica quanti ulteriori caratteri esistono nel nome.

Tab. F-1 Allocazione di Variabili (cont.)

SPIAZZAMENTO	LUNGHEZZA	DESCRIZIONE	COMMENTI
			- I byte con spiazzamento da 4 a n contengono i caratteri supplementari del nome. Tre byte vengono riservati per il nome di variabile, anche se la lunghezza del nome è di uno o due caratteri. In questo caso il dato (o il descrittore di stringa) inizia con spiazzamento 4.
4+n	2	Valore intero	Gli interi sono memorizzati con il byte di ordine inferiore prima e quello di ordine superiore dopo.
	3	Descrittore di stringa	- Il primo byte contiene la lunghezza della stringa (0-255). - Il secondo byte è il byte di ordine inferiore dello spiazzamento dall'origine del Segmento Dati GW-BASIC. - Il terzo byte è il byte di ordine superiore dello spiazzamento dall'origine del Segmento Dati GW-BASIC.
	4	Valore in semplice precisione	Nel formato a virgola mobile.

Tab. F-1 Allocazione di Variabili (cont.)

SPIAZZAMENTO	LUNGHEZZA	DESCRIZIONE	COMMENTI
	8	Valore in doppia precisione	Nel formato a virgola mobile.

Tab. F-1 Allocazione di Variabili

RAPPRESENTAZIONE INTERNA DEI NUMERI A VIRGOLA MOBILE

La sezione che segue descrive la rappresentazione interna dei numeri in GW-BASIC.

Semplice Precisione - Mantissa a 24 bit

0	1	2		3
loman		S	himan	exp

dove:
loman è la parte bassa della mantissa
S è il segno
himan è la parte alta della mantissa
exp è l'esponente
man è himan:...:loman

Se exp = 0, allora number = 0

Se exp < > 0, allora la mantissa viene normalizzata e
number = sgn* .1 man *2** (exp-80h)

In semplice precisione (notazione esadecimale-byte da quello di ordine inferiore a quello di ordine superiore):

00000080 = .5
00008080 = -.5

Doppia precisione - Mantissa a 56 bit

0	1	2	3	4	5	6	7
loman						S himan	exp

FILE CONTROL BLOCK

La tabella riportata di seguito illustra le informazioni contenute nel File Control Block (FCB). Gli spiazzamenti sono relativi all'indirizzo restituito dalla funzione VARPTR. Il File Control Block di GW-BASIC è diverso da quello di MS-DOS.

SPIAZZAMENTO	LUNGHEZZA	NOME	SIGNIFICATO
0	1	Modo	Modo in cui il file è stato aperto: 1 - Input 2 - Output 4 - Accesso diretto 16 - Append 32 - Riservato 128 - Riservato
1	38	FCB	MS-DOS File Control Block.

Tab. F-2 File Control Block (cont.)

SPIAZZAMENTO	LUNGHEZZA	NOME	SIGNIFICATO
39	2	CURLOC	Numero di settori letti o scritti (per file sequenziali); numero dell'ultimo record letto o scritto + 1 (per file ad accesso diretto).
41	1	ORNOFS	Numero di byte per settore.
42	1	NMLOFS	Numero di byte trasferiti nel buffer di INPUT.
43	3	—	Riservato.
46	1	DEVICE	Numero del device: 0-9 Dischi (A: - J:) 255 Tastiera 254 SCRN: 253 LPT1: 252 CAS1: 251 COM1: 250 COM2: 249 LPT2: 248 LPT3:
47	1	WIDTH	Ampiezza di linea del device.
48	1	POS	Posizione nel buffer per PRINT#.

Tab. F-2 File Control Block (cont.)

SPIAZZAMENTO	LUNGHEZZA	NOME	SIGNIFICATO
49	1	FLAGS	Uso interno durante l'esecuzione di BLOAD/BSAVE. Non utilizzato per file dati.
50	1	OUTPOS	Posizione di output utilizzata in operazioni di tabulazione.
51	128	BUFFER	Buffer Dati, usato per trasferimento dati tra MS-DOS e GW-BASIC. Si deve usare questo spiazzamento per esaminare i dati in I/O sequenziale.
179	2	URECL	Lunghezza record per file ad accesso diretto, stabilita nell'istruzione OPEN. Valore di default 128.
181	2	PHYREC	Numero del record fisico corrente.
183	2	LOGREC	Numero del record logico corrente.
185	1	—	Riservato.
186	2	OUTPOS	Posizione utilizzata da PRINT#, INPUT# e WRITE#. Solo per file su disco.

Tab. F-2 File Control Block (cont.)

SPIAZZAMENTO	LUNGHEZZA	NOME	SIGNIFICATO
188	n	FIELD	Field Buffer corrente. La sua lunghezza, in byte, viene stabilita dall'opzione /S: del comando GWBASIC. VRECL byte vengono trasferiti tra BUFFER e FIELD nelle operazioni di I/O. Si deve usare questo spiazzamento per esaminare i dati di una operazione di I/O relativa a file ad accesso diretto.

Fig. F-2 File Control Block

MAPPA DELLA MEMORIA

La mappa riportata di seguito illustra la struttura della memoria del sistema.

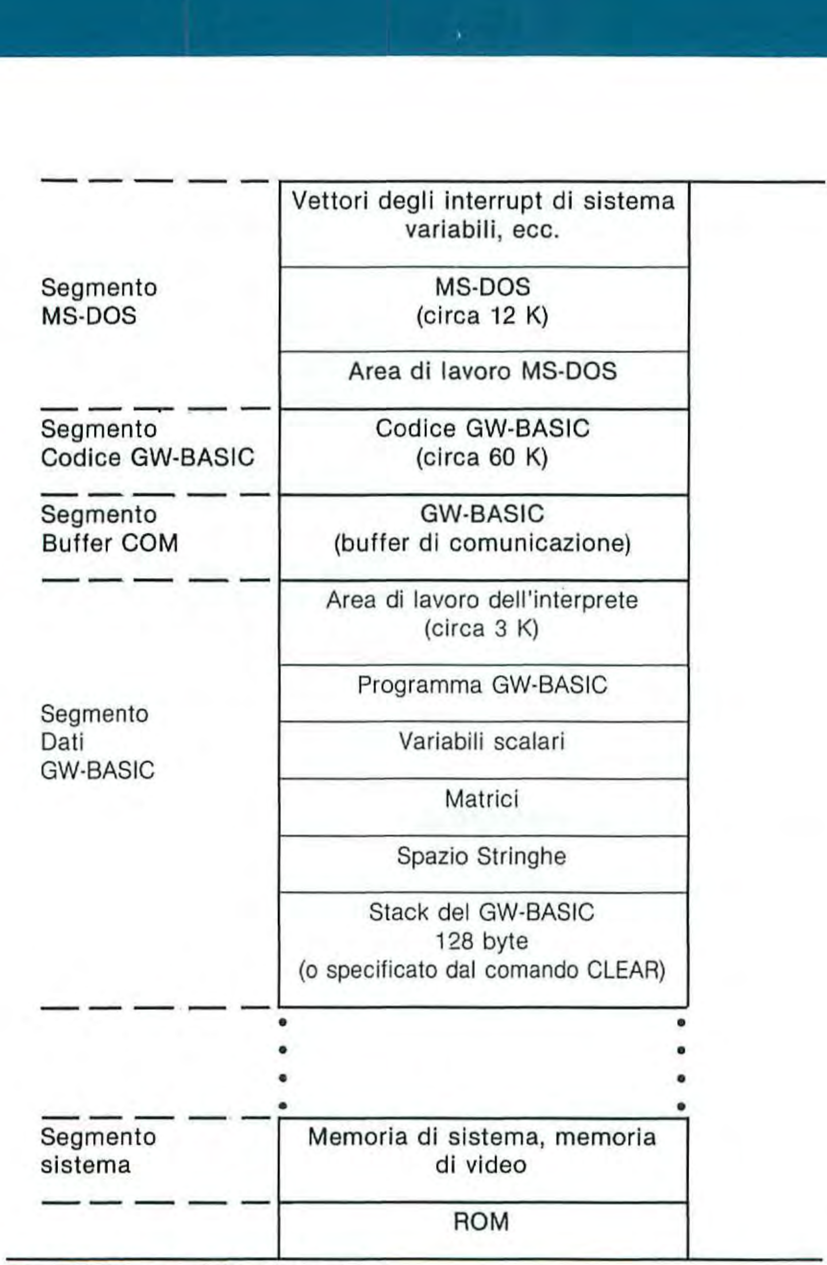


Fig. F-3 Mappa della Memoria

Nell'area di lavoro GW-BASIC ci sono alcune variabili a disposizione dell'utente, alle quali si può accedere tramite PEEK o POKE. Queste variabili non sono agli indirizzi effettivi specificati: PEEK o POKE mappano gli indirizzi specificati nelle locazioni effettive.

L'indirizzo iniziale del Segmento Dati GW-BASIC può essere ottenuto esaminando la parola situata all'indirizzo assoluto 0:510Hex (questo è un indirizzo effettivo).

Non si richiede quest'informazione per l'esecuzione di PEEK o di POKE. Un'istruzione DEF SEG specifica il segmento giusto. In questo caso, le locazioni che seguono hanno il seguente significato:

INDIRIZZO DECIMALE	SPIAZZAMENTO ESADECIMALE	SIGNIFICATO
46	2E	Numero di linea corrente
839	347	Numero di linea corrente dell'errore
48	30	Spiazzamento di inizio del programma
856	358	Spiazzamento di inizio delle variabili
106	6A	Locazione PEEK/POKE del buffe di tastiera

Le prime quattro locazioni sono delle variabili di tipo word che vengono mappate agli indirizzi dati quando vengono utilizzate PEEK e POKE. Sono variabili a 2-byte; perciò, l'esecuzione della linea di programma:

```
1234 X = PEEK(46) + 256 * PEEK(47)
```

imposta X = 1234, che è il numero della linea corrente.

Impostando 0 con l'istruzione POKE alla locazione 106 (decimale) si cancella il buffer di linea di GW-BASIC, il quale memorizza i caratteri ricevuti dal buffer di tastiera. Questi caratteri sono gestiti da Screen Editor.

Qualsiasi altro valore scritto con POKE a questo indirizzo non ha alcun effetto.

PEEK(106) restituisce 0 se nessun carattere della tastiera è disponibile ed 1 se è disponibile qualche carattere.

G. GLOSSARIO

SOMMARIO

Questa appendice contiene il glossario dei termini GW-BASIC, parole speciali e definizioni usate all'interno di questo manuale.

accesso diretto: Metodo di accesso ai file in cui ogni record è indirizzabile direttamente. Possibilità di leggere o scrivere informazioni su qualsiasi locazione all'interno di un device di memorizzazione dati.

accesso sequenziale: Metodo di accesso ai file in cui i record vengono elaborati in ordine sequenziale, cioé vengono letti nello stesso ordine in cui sono stati scritti.

animazione: Tecnica che consiste nel visualizzare una sequenza di immagini, in modo da vedere gli oggetti in movimento sul video.

argomento: Valore passato dal programma principale a una funzione o a un sottoprogramma.

argomento fittizio: Parametro fittizio in una funzione, un'istruzione o un comando. Deve essere impostato un valore, ma GW-BASIC lo ignora.

aspect ratio: È una frazione atta a specificare quanti pixel sono necessari per avere un segmento di ugual misura lungo i due assi. Il numeratore specifica il numero di pixel lungo l'asse Y e il denominatore il numero di pixel lungo l'asse X.

baud rate: Velocità di trasmissione, sinonimo, in effetti, di numero di segnali (di solito bit) al secondo.

boot: Caricamento di un sistema operativo in memoria.

campo: In un record, un'area specifica usata per un tipo particolare di dati.

carattere di definizione di tipo: Carattere speciale impostato alla fine di una variabile. Può essere: % (variabile intera), ! (variabile in semplice precisione), # (variabile in doppia precisione) o $ (variabile stringa).

carattere jolly: Simbolo speciale usato per rappresentare qualsiasi carattere singolo (?) o stringa di caratteri (*) in un nome di file.

chiamata: Salto o trasferimento del controllo a un sottoprogramma specificato.

ciclo: Insieme di istruzioni eseguite ripetutamente un dato numero di volte.

clausola: Parte opzionale di un'istruzione o di un comando che normalmente comprende una parola chiave e uno o più parametri.

clipping: Le istruzioni grafiche usano il clipping, cioé tutti i punti che cadono fuori dai margini della finestra fisica corrente sono ignorati.

codice di colore: Numero compreso tra 0 e 15 che identifica un colore.

codice esteso: Tramite la funzione INKEY$ viene restituito il codice ASCII esteso, se viene digitato un tasto (o una combinazione di tasti) che non può venire associato a un codice ASCII standard (vedere "MS-DOS Guida Utente").

colore di primo piano: Il colore di visualizzazione dei caratteri (colore di primo piano per il testo) o il colore usato per tracciare punti e linee (colore di primo piano per la grafica).

colore di fondo: Colore dell'area che circonda il soggetto. In particolare, è il colore di fondo che circonda un carattere (colore di fondo per il testo) o il colore di riempimento quando viene cancellata una finestra di grafica (colore di fondo per la grafica).

commento: Istruzione usata per documentare un programma, che, in GW-BASIC, può essere impostata tramite REM o un'apice (') seguito dalla stringa di commento. L'apice permette anche di inserire commenti alla fine di una linea GW-BASIC.

concatenamento: Operazione che unisce due stringhe con l'uso dell'operatore " + ".

coordinate: Numeri che identificano una posizione su video. Possono essere coordinate di testo per identificare un carattere o il cursore (espresse in termini di righe e colonne), oppure coordinate cartesiane *x* e *y*.

coordinate assolute: In grafica, le coordinate cartesiane di un punto espresse secondo il sistema di coordinate in vigore.

coordinate fisiche: In grafica, le coordinate cartesiane di un punto del video espresse in pixel.

coordinate logiche: In grafica, le coordinate cartesiane di un punto del disegno da visualizzare espresse secondo il sistema di coordinate scelte dall'utente.

coordinate relative: Nella grafica, i valori di *x* e *y* che identificano la posizione di un punto in funzione della posizione del punto corrente.

costante: Valore che non cambia durante l'esecuzione di un programma. Una costante può essere di tipo stringa o numerica e in quest'ultimo caso può essere un numero intero, in semplice o doppia precisione.

cursore: Marcatore mobile usato per indicare la posizione del prossimo carattere su video. Ci sono tre tipi di cursore: quello di scrittura, quello di inserimento e quello utente. La forma e la frequenza di lampeggiamento dei cursori di scrittura e utente sono programmabili. Il cursore utente di norma non è visibile, ma può esserlo con l'istruzione LOCATE.

debug: Relativo all'individuazione e alla correzione d'errori in un programma.

default: Valore di un parametro che viene assunto dal sistema quando non viene fornito alcun valore specifico.

delimitatore di stringa: Carattere che delimita una stringa di caratteri e che quindi non può far parte della stringa.

device: Periferica integrata o non integrata. Esempi di device sono la stampante, la tastiera, il video, una porta di I/O, ecc.

directory: Contiene i nomi dei file su disco, insieme alle informazioni che indicano a MS-DOS dove trovare il file.

directory corrente: Directory su cui si sta lavorando. È possibile cambiare la directory corrente tramite il comando CHDIR. Subito dopo la formattazione la directory corrente è Root.

disco: Termine generico per specificare un hard-disk o un diskette.

diskette: Floppy disk da 5 pollici e 1/4.

doppia precisione: La massima precisione che GW-BASIC può gestire. Se un numero contiene più di 7 cifre è un numero in doppia precisione.

drive: Sinonimo di drive su disco. Può essere specificato da A: (primo drive di diskette), B: (secondo drive di diskette), C: (primo drive di hard-disk), ecc.

drive di default: Quando viene dato uno specificatore di file o di pathname, il drive di default rappresenta il default in MS-DOS prima di passare in ambiente GW-BASIC. Durante le procedure di inizializzazione il drive di default è quello da cui viene caricato MS-DOS.
Il drive di default può essere cambiato in MS-DOS solo impostando lo specificatore di drive (A:, B:, C:, ecc.) prima del nome di un comando MS-DOS, o da solo.

edit: Per modificare una linea GW-BASIC visualizzata.

effetto zoom: In grafica, un particolare ingrandito di un disegno.

end of file (EOF): Marcatore che segue immediatamente l'ultimo record di un file.

espressione: Algoritmo che restituisce un valore numerico (espressioni numeriche, di confronto o logiche) o un concatenamento di stringhe che restituiscono un valore stringa (espressione stringa).

espressione logica: Consiste di un operando preceduto dall'operatore logico NOT, di due operandi separati da un altro operatore logico (AND, OR, EQV e IMP) o di due operandi separati da un operatore logico e NOT. Gli operandi possono essere espressioni numeriche o di confronto. Entrambi hanno valori numerici. Il risultato di un'espressione logica è numerico ed è un'intero compreso nell'intervallo da -32768 a 32767.

espressione numerica: Espressione il cui calcolo restituisce un valore numerico, che può essere intero, in semplice o doppia precisione.

espressione relazionale: Espressione che confronta due espressioni numeriche o stringa per mezzo di un operatore di confronto. Un'espressione relazionale può essere considerata un caso particolare di un'espressione numerica, poiché il risultato è numerico: 1 (se la relazione è vera), 0 (se la relazione è falsa).

espressione stringa: Espressione che restituisce un valore stringa.

fattore di scala: In grafica, numero reale che moltiplicato per la distanza specificata dai comandi U, D, R, L, E, F, G, H, dell'istruzione DRAW, fornisce lo spostamento effettivo in pixel.

file: Insieme di record. È possibile accedere ai record di un file in modo sequenziale o in modo diretto (tramite il numero del record).

finestra di testo: In grafica, sezione rettangolare del video in cui viene visualizzato un testo. Può essere definita da un'istruzione VIEW PRINT o WIDTH.

finestra fisica: In grafica, sezione rettangolare del video, definita dall'istruzione VIEW, sulla quale può essere disegnata una figura. È possibile avere più finestre fisiche su video, ma una sola è quella corrente.

finestra logica: In grafica, sezione rettangolare dello spazio, definita dall'istruzione WINDOW, i cui punti vengono proiettati sui punti della finestra fisica o sull'intero video.

floppy disk: Disco flessibile.

folding di linea: Continuazione di una linea logica su una linea fisica seguente, in modo da poter modificare la linea (inserendo nuovi caratteri o cancellandone alcuni) senza perdere altri caratteri su quella linea.

full duplex: Sistema di comunicazione che permette operazioni simultanee in entrambe le direzioni.

funzione: Algoritmo che restituisce un unico valore. Una funzione può essere di tipo utente o intrinseca. Può essere richiamata specificandone semplicemente il nome, seguito (tra parentesi) da uno o più argomenti che rappresentano i valori che i parametri della funzione devono assumere.

funzione built-in: Vedere "funzione intrinseca".

funzione utente: Funzione che l'utente deve definire prima di richiamarla.

funzione intrinseca: Funzione che l'utente può richiamare senza definirla, poiché è parte di GW-BASIC (per esempio SIN (x)).

garbage collection: Compressione dello spazio di memoria riservato alle stringhe effettuate eliminando gli spazi non utilizzati. GW-BASIC esegue la compressione delle stringhe durante l'esecuzione della funzione FRE o quando tutta la memoria libera è stata utilizzata.

GW-BASIC: In questo manuale il termine GW-BASIC si riferisce a GW-BASIC Microsoft, così come viene implementato sui Personal Computer Olivetti.

half duplex: Sistema di comunicazione che permette operazioni in entrambe le direzioni, ma non simultaneamente.

hard disk: Disco rigido. In questo manuale si riferisce a un disco di tipo Winchester di 5 pollici e 1/4.

indice: Numero intero positivo che identifica la posizione di un elemento in una matrice.

indirizzo: Locazione di memoria.

infinito di macchina: Il numero più alto che può essere rappresentato in formato interno.

interruzione: Sospensione di una elaborazione, come l'esecuzione di un programma, causata da un evento esterno all'elaborazione, in modo tale che l'elaborazione può essere ripresa.

istruzione: Istruzione che comunica al computer di svolgere una sequenza di operazioni.

linea: Una linea GW-BASIC può iniziare con un numero di linea (se è una linea di programma) o con una lettera (se è una linea immediata). La linea può contenere una o più istruzioni o comandi GW-BASIC (separati da due punti) e può essere lunga fino a 255 caratteri.

linea corrente: Vedere "punto corrente".

linea diretta: Vedere "linea immediata".

linea immediata: Linea GW-BASIC che inizia con una lettera. Viene eseguita non appena premuto **CR**.

linea indiretta: È una linea GW-BASIC che inizia con un numero di linea.

loop: Esecuzione ripetuta di una serie di istruzioni per un dato numero di volte.

mantissa: Il numero che non è l'esponente nella notazione a virgola mobile.

mappa di bit: Area di memoria i cui bit sono associati ai punti del video. Se un solo bit è associato a un punto del video si possono visualizzare due colori (B/N), se due bit sono associati a un punto del video si possono visualizzare quattro colori.

maschera del mosaico: In grafica, struttura rettangolare di bit utilizzata in modo ripetitivo per riempire un'area con una data configurazione di colori.

matrice: Lista di variabili dello stesso tipo e con lo stesso nome. La posizione di ogni variabile (o elemento) nella lista è data dal valore dei suoi indici.

modo di accesso: Tecnica usata per indirizzare un record specifico in un file.

modo diretto: Vedere "modo immediato".

modo immediato: Questo modo viene usato per impostare ed eseguire immediatamente una linea GW-BASIC.

modo indiretto: Vedere "modo programma".

modo programma: Modo usato per introdurre in memoria una linea di programma GW-BASIC. Per comunicare a GW-BASIC che la linea introdotta fa parte di un programma, è necessario iniziare la linea con un numero. Quando viene premuto **CR**, la linea di programma viene caricata in memoria ma non viene eseguita. Le linee vengono memorizzate in sequenza, così da formare un programma GW-BASIC. Per eseguire il programma, premere **RUN CR**.

MS-DOS: Microsoft-Disk Operating System. In questo manuale si riferisce solo al Sistema Operativo su disco per il Personal Computer Olivetti.

nome di file: Nome assegnato a un file. Può comprendere un'estensione di tre caratteri separati dal nome di file con un punto. (Le estensioni BAS, BAT, COM o EXE assumono significati speciali.)

numero di colore: In grafica, numero (compreso tra 0 e 3) che seleziona un colore sulla tavolozza attuale.

numero di file: numero (compreso tra 1 e 255) associato a un file al momento della sua apertura. Viene usato da qualsiasi istruzione di I/O per fare riferimento al file.

offset: Vedere "spiazzamento".

overflow: In una operazione aritmetica, la generazione di un valore che va oltre la capacità del registro o della locazione che deve ricevere il risultato.

overlay: I programmi troppo grandi per essere residenti in memoria possono essere divisi in segmenti logici (o overlay) ciascuno

dei quali viene trasferito in memoria, eseguito e quindi cancellato per permettere il trasferimento in memoria di un altro overlay.

pagina attiva: Sezione del buffer del video su cui è possibile scrivere tramite istruzioni di output. In Modo Testo essa può essere diversa dalla pagina visiva.

pagina visiva: Sezione del buffer del video correntemente visualizzata. Può essere diversa dalla pagina attiva in Modo Testo.

parametro: Valore scelto dall'utente impostando un comando o un'istruzione. Sinonimo di argomento. Un parametro attuale è un valore che sostituisce un parametro formale in una data procedura o in una data funzione, quando vengono richiamate.

parola chiave: Parola che identifica un'istruzione, una funzione o un comando. È una parola riservata, cioé non può essere usata come nome di una variabile.

pathname: Sequenza di nomi di directory separate dalla barra rovesciata (\). L'ultimo elemento di un pathname deve essere una directory o un nome di file. Nel primo caso il pathname è usato per specificare una directory; nel secondo caso per specificare un file. Il primo elemento di un pathname deve essere un nome di device.

pixel: Punto indirizzabile sul video tramite le sue coordinate.

porta RS-232-C: Buffer per il trasferimento dati da/a altro computer o periferica.

programma corrente: Programma residente in memoria.

prompt: Messaggio che compare sul video per ricordare all'utente di svolgere un'azione specifica.

punto corrente: In grafica, è l'ultimo punto a cui si è fatto riferimento.

reboot: Ricaricare un sistema operativo da disco in memoria.

record: Gruppo di uno o più campi consecutivi logicamente collegati. Un file è un insieme di record.

record a lunghezza fissa: Elementi numerabili di un file, tutti della stessa lunghezza. Per esempio, in GW-BASIC i file ad accesso diretto hanno record a lungheza fissa.

record a lunghezza variabile: Record la cui lunghezza non dipende da quella di altri record del file.

rilevamento di errori: Quando si verifica un errore il controllo del programma può essere trasferito automaticamente a una routine di gestione degli errori scritta dall'utente invece che alla routine standard. Questa routine individua tutti gli errori che l'utente desidera gestire e indica il procedimento da seguire in ogni singolo caso.

rilevamento di eventi: Quando si verifica un certo evento il controllo del programma può essere trasferito a una routine di rilevamento scritta dall'utente. Questa, dopo avere gestito l'evento, esegue un'istruzione RETURN che fa riprendere l'esecuzione del programma. Gli eventi che possono venire gestiti sono: la ricezione di caratteri in una porta RS-232-C, la pressione di un dato tasto, la scadenza di un dato intervallo di tempo e lo svuotamento della coda di musica di sottofondo.

rivestimento: Riempimento dell'area, interna o esterna, di una figura con una struttura di colori (mosaico).

scan code: Codice posizionale di un tasto che dipende dalla posizione fisica del tasto ed è indipendente dalla tastiera nazionale usata.

scanline: Linea orizzontale di pixel su video.

scroll: Spostamento su video di un testo o di una sua parte, verticalmente o orizzontalmente, così da visualizzare i caratteri che altrimenti non compaiono.

segmento: Area di memoria di 64K byte.

segmento corrente: Segmento di memoria, definito dall'ultima istruzione DEF SEG eseguita, oppure il Segmento Dati GW-BASIC.

semplice precisione: Se un numero non è intero e contiene 7 cifre o meno di 7 cifre, è un numero in semplice precisione.

soft key: Sinonimo di tasto funzione.

soft key display: Visualizzazione sulla 25 linea del video dei valori associati ai tasti funzione.

sottoprogramma: Sezione di un programma GW-BASIC richiamata da un'istruzione GOSUB oppure ON...GOSUB. Alla fine dell'esecuzione di un sottoprogramma il controllo viene restituito alla prima istruzione che segue l'ultima GOSUB (o ON...GOSUB) eseguita.

specificatore di file: Unico identificatore di un file, comprende il nome di file eventualmente preceduto da uno specificatore di device (A:, B:, C:,...). Se non viene specificato un device, viene assunto il drive corrente.

spiazzamento (offset): Distanza in byte da una data locazione di memoria (normalmente l'origine del segmento corrente).

stack: Area di memoria in cui i dati vengono contenuti provvisoriamente, in modo tale che l'ultimo dato inserito è il primo ad essere estratto.

stato comandi: GW-BASIC è in stato comandi quando su video compare Ok, cioé quando GW-BASIC è in attesa della digitazione di una linea immediata o di un programma da parte dell'utente.

stringa nulla: Una stringa di lunghezza uguale a zero, cioé senza caratteri (viene rappresentata come "").

subroutine: Vedere "sottoprogramma".

switch: Una della opzioni del comando GW-BASIC, specificata con una barra (/) seguita da una lettera e due punti.

tastiera alfanumerica: Sezione centrale della tastiera usata come la tastiera di una normale macchina per scrivere.

tastiera numerica: Sezione destra della tastiera dedicata a numeri, simboli aritmetici, tasti di movimento cursore e alcuni caratteri di controllo.

tasto di chiusura impostazione o di ritorno a capo (CR): Carattere che permette di spostare la posizione di stampa o di visualizzazione alla prima posizione della linea successiva. Impostando CR alla fine di una linea, questa viene passata a GW-BASIC per l'elaborazione.

tasto funzione: Tasto a cui l'utente può assegnare un significato speciale. Premuto il tasto è possibile generare qualsiasi stringa di caratteri. Alcuni tasti funzione possono essere già assegnati al momento dell'inizializzazione.

tavolozza: In grafica, insieme dei colori di primo piano ordinati per numero di colore.

tipo di un'espressione: È il tipo di dato (stringa, intero, in semplice o doppia precisione) che costituisce il risultato dell'espressione. Dipende dal tipo di operandi dell'espressione.

tipo di una variabile: Indica se una variabile è di tipo stringa o numerico. Se numerica indica anche se è intera, in semplice o doppia precisione. Il tipo di una variabile può essere definito tramite l'istruzione DEF (INT, SNG, DBL o STR) o tramite un carattere di definizione alla fine del nome di variabile.

trapping: Forma speciale di interruzione condizionata, attivata da un evento che deve essere rilevato. Si riferisce anche all'azione da intraprendere dopo il rilevamento.

trasmissione asincrona: Metodo per la trasmissione di dati in cui ogni carattere trasmesso è preceduto da un bit di start e seguito da un bit di stop, pemettendo così di variare l'intervallo di tempo tra i caratteri.

valore booleano: Valore numerico che GW-BASIC interpreta (in un'istruzione o in un comando) come "vero" o "falso" per dirigere il flusso del programma o per altri scopi.

variabile: Nome di un dato il cui valore può cambiare durante l'esecuzione del programma. Una variabile può essere una variabile semplice o un elemento di vettore. Il nome di una variabile specifica anche il suo tipo (numerico o stringa).

variabile di destinazione: Variabile alla sinistra del segno di uguale in un'istruzione di assegnazione.

variabile numerica: Variabile semplice o elemento di vettore il cui valore è numerico; cioé intero, in semplice o doppia precisione, a seconda del tipo definito per la variabile.

variabile stringa: Variabile semplice o elemento di vettore il cui valore è una stringa.

vettore: Matrice ad una dimensione.

AVVISO

La Ing. C. Olivetti & C., S.p.A. si riserva il diritto di apportare modifiche in qualsiasi momento e senza preavviso.

Il presente manuale è concesso in uso al Cliente alle condizioni specificate nella Licenza d'uso acclusa al Programma cui il manuale si riferisce.

Code 4001810 Z (3)
Printed in Italy

olivetti
PERSONAL
COMPUTER

olivetti